



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

ULB

# **Semantically Enhanced and Minimally Supervised Models for Ontology Construction, Text Classification, and Document Recommendation**

Alkhatib, Wael  
(2020)

DOI (TUprints): <https://doi.org/10.25534/tuprints-00011890>

Lizenz:



CC-BY-ND 4.0 International - Creative Commons, Namensnennung, keine Bearbeitung

Publikationstyp: Dissertation

Fachbereich: 18 Fachbereich Elektrotechnik und Informationstechnik

Quelle des Originals: <https://tuprints.ulb.tu-darmstadt.de/11890>

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

SEMANTICALLY ENHANCED AND MINIMALLY SUPERVISED MODELS  
for Ontology Construction, Text Classification, and Document Recommendation

Dem Fachbereich Elektrotechnik und Informationstechnik  
der Technischen Universität Darmstadt  
zur Erlangung des akademischen Grades eines  
Doktor-Ingenieurs (Dr.-Ing.)  
genehmigte Dissertation

von

WAEL ALKHATIB, M.SC.

Geboren am 15. October 1988 in Hama, Syrien

Vorsitz: Prof. Dr.-Ing. Jutta Hanson  
Referent: Prof. Dr.-Ing. Ralf Steinmetz  
Korreferent: Prof. Dr.-Ing. Steffen Staab

Tag der Einreichung: 28. January 2020  
Tag der Disputation: 10. June 2020

Hochschulkennziffer D17  
Darmstadt 2020

This document is provided by tuprints, e-publishing service of the Technical University Darmstadt.

<http://tuprints.ulb.tu-darmstadt.de>

[tuprints@ulb.tu-darmstadt.de](mailto:tuprints@ulb.tu-darmstadt.de)

Please cite this document as:

URN:nbn:de:tuda-tuprints-118909

URL:<https://tuprints.ulb.tu-darmstadt.de/id/eprint/11890>

This publication is licensed under the following Creative Commons License:

*Attribution-No Derivatives 4.0 International*

<https://creativecommons.org/licenses/by-nd/4.0/deed.en>

Wael Alkhatib, M.Sc.: *Semantically Enhanced and Minimally Supervised Models*,  
for Ontology Construction, Text Classification, and Document Recommendation  
© 28. January 2020

SUPERVISORS:

Prof. Dr.-Ing. Ralf Steinmetz

Prof. Dr.-Ing. Steffen Staab

LOCATION:

Darmstadt TIME FRAME:

28. January 2020

## ABSTRACT

---

The proliferation of deliverable knowledge on the web, along with the rapidly increasing number of accessible research publications, make researchers, students, and educators overwhelmed. Linked data platforms like *SciGraph*<sup>1</sup> reduce this information overload by combining data from heterogeneous information sources and link them to ontologies that describe how these resources are related. Linked data platforms provide functionalities to improve the accessibility and discoverability of these resources. These functionalities include methods for maintaining and updating the ontologies used, for the assignment of concepts to resources as well as for providing recommendations of relevant resources. About 80% of information sources on the Internet originate in form of unstructured content. This triggers the need for automated methods that leverage the wealth of information embedded in unstructured content to realize the needed functionalities.

This thesis provides contributions concerning three building blocks of the construction of linked data platforms from unstructured information sources, namely ontology construction and enrichment, text classification, and document recommendation. The majority of Machine Learning (ML) methods used for studying these problems are characterized by the intensive reliance on complicated feature engineering, which is a tedious, time consuming, and domain-specific process. Our work is motivated by the potential of using lexical-semantic resources and deep learning to address the research challenges in the current approaches. On the one side, existing lexical-semantic resources encode various types of information about words such as their meaning and semantic relations. On the other side, deep learning methods have achieved state-of-the-art performance on challenging Natural Language Processing (NLP) problems, i.e., text classification and semantic relation extraction. The rise of distributed representations is the key to the breakthrough of deep learning on various NLP tasks. The focus of this work is to develop, implement, and evaluate new approaches that better leverage the semantic similarities and regularities between words in large text corpora to minimize the hand-crafted feature engineering in current approaches.

With regard to ontology construction and enrichment, we present Onto.KOM: a minimally supervised ontology learning system that uses unstructured text as input in addition to existing lexical databases. We study the effectiveness of using our approach for semantic relation classification regarding different influencing aspects, namely the input representation, the deep network structure used, and the types of semantic relations.

In the scope of multi-label text classification, our contributions lie under three main areas: First, we propose an approach for feature selection using the typed dependencies between words as a measure to select the most essential features. We compare our approach with multiple statistical and semantic-based techniques, to investigate the

---

<sup>1</sup> <https://www.springernature.com/gp/researchers/scigraph>



advantage of leveraging the semantic and syntactic relationships between words to improve the quality of selected features. Second, we analyse the performance of deep learning structures on a small dataset of long documents where traditional techniques tend to perform better. Besides, we develop a new model that uses the distributed representations of document fragments and deep learning structures. We compare the new model with a wide range of feature selection and text classification techniques. Third, we address the label imbalance problem and the lack of sufficient training samples. In this scope, we develop a training-less classifier based on lexical-semantic resources as a base for classification. We transform the classification problem into graph matching problem.

Concerning the recommendation of relevant resources, we address the problem of citation recommendation as a particular use case of document recommendation. We propose two models for combining the different heterogeneous information sources, such as the content of papers, co-authorship information, and previously cited papers to provide personalized citation recommendation.

## KURZFASSUNG

---

Die zunehmende Anzahl im Internet von verfügbaren Ressourcen und die schnell ansteigende Zahl von zugänglichen Forschungspublikationen überfordern Forscher, Studenten und Dozenten. Linked Data Plattformen wie *SciGraph* reduzieren diese Informationsüberflutung, indem sie Ressourcen aus den heterogenen Quellen im Internet kombinieren und mit Ontologien verknüpfen. Ontologien beschreiben, wie Konzepte und damit verlinkte Ressourcen zusammenhängen. Linked Data Plattformen sollen Funktionen bereitstellen, um die Zugänglichkeit und Auffindbarkeit der Ressourcen zu verbessern. Diese Funktionen umfassen Methoden zur Wartung und Aktualisierung der verwendeten Ontologien, zur Zuordnung von Konzepten zu Ressourcen sowie zur Abgabe von Empfehlungen zu relevanten Ressourcen. Etwa 80% der Ressourcen im Internet kommen in Form von unstrukturierten Inhalten vor. Dies führt einem Bedarf an automatisierten Methoden, um die Fülle der in unstrukturierten Inhalten eingebetteten Informationen zu nutzen, und die benötigten Funktionen zu realisieren.

Diese Arbeit beinhaltet Beiträge zu drei Aufgaben zum Aufbau von Linked Data Plattformen aus unstrukturierten Informationsquellen, nämlich Ontologiegenerierung und -anreicherung, Textklassifizierung und Dokumentenempfehlung. Die Mehrheit der Methoden des maschinellen Lernens (ML), die bisher zur Lösung dieser Aufgaben verwendet werden, zeichnen sich durch die intensive Abhängigkeit von einem kompliziertem Feature Engineering aus. Das ist ein langwieriger, zeitaufwendiger und domänenspezifischer Prozess. Die Motivation unserer Arbeit liegt in dem Potenzial, lexikalisch-semantische Ressourcen und Deep Learning zur Bewältigung der Forschungsherausforderungen in den drei oben genannten Aufgaben zu nutzen. Auf der einen Seite kodieren die vorhandenen lexikalisch-semantischen Ressourcen verschiedene Arten von Informationen über Wörter wie ihre Bedeutung und ihre semantischen Beziehungen. Auf der anderen Seite haben Deep-Learning-Methoden bei anspruchsvollen NLP-Problemen, d.h. Textklassifizierung und semantische Beziehungsextraktion, Spitzenleistungen erzielt. Der Entwicklung der Distributed Representation ist der Schlüssel zum Durchbruch des Deep Learning bei verschiedenen NLP-Aufgaben. Der Schwerpunkt dieser Arbeit liegt auf der Entwicklung, Implementierung und Bewertung neuer Ansätze, die die semantischen Ähnlichkeiten und Regelmäßigkeiten zwischen Wörtern in großen Textkorpora besser nutzen, um das manuelle Feature Engineering in aktuellen Lösungsansätzen zu minimieren.

Im Hinblick auf die Generierung und die Anreicherung von Ontologien stellen wir Onto.KOM vor: ein minimal überwachtes Ontologie-Lernsystem, das neben bestehenden lexikalischen Datenbanken auch unstrukturierten Text als Eingabe verwendet. Wir untersuchen die Effektivität der Verwendung unseres Ansatzes zur semantischen Beziehungsklassifizierung hinsichtlich verschiedener Einflussaspekte, nämlich der Ein-

gangsdarstellung, der verwendeten Netzwerkstruktur und der Arten semantischer Beziehungen.

Im Rahmen der Klassifizierung von Multilabel-Texten liegen die Beiträge in drei Hauptbereichen: Zuerst schlagen wir einen Ansatz für die Merkmalsauswahl vor, bei dem die typisierten Abhängigkeiten zwischen Wörtern als Maß für die Auswahl der wichtigsten Merkmale verwendet werden. Wir vergleichen unseren Ansatz mit mehreren statistischen und semantischen Techniken, um den Vorteil der Nutzung der semantischen und syntaktischen Beziehungen zwischen Wörtern zur Verbesserung der Qualität ausgewählter Merkmale zu untersuchen. Zweitens, analysieren wir die Leistung von Deep Learning Strukturen anhand eines kleinen Datensatzes langer Dokumente, in dem traditionelle Techniken tendenziell besser abschneiden. Außerdem entwickeln wir ein neues Modell, das die verteilten Darstellungen von Dokumentenfragmenten und Deep Learning Strukturen nutzt. Wir vergleichen das neue Modell mit einer Vielzahl von Techniken zur Merkmalsauswahl und Textklassifizierung. Drittens, geht es um das Problem der Label Imbalance und des Fehlens ausreichender Trainingsdaten. In diesem Rahmen entwickeln wir einen traininglosen Klassifikator, der auf lexikalisch-semantischen Ressourcen als Grundlage für die Klassifizierung basiert. Wir umwandeln das Klassifikation Problem in einen Graph Matching Problem.

Hinsichtlich der Empfehlung relevanter Ressourcen gehen wir auf das Problem der Zitatempfelung als besonderen Anwendungsfall der Dokumentenempfehlung ein. Wir schlagen zwei Modelle für die Kombination verschiedener heterogener Informationsquellen vor, wie z.B. den Inhalt von Publikationen, Co-Autoreninformationen und zuvor zitierte Papers für die Bereitstellung einer personalisierten Zitatempfelung.

## CONTENTS

---

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Research Challenges . . . . .	3
1.3	Research Goals and Contributions . . . . .	4
1.4	Structure of the Thesis . . . . .	6
<b>2</b>	<b>FOUNDATIONS</b>	<b>7</b>
2.1	Ontology Learning and Lexical-Semantic Resources . . . . .	7
2.1.1	Tasks of Ontology Learning . . . . .	7
2.1.2	Lexical-Semantic Resources . . . . .	9
2.2	Natural Language Processing . . . . .	10
2.2.1	Text Preprocessing Tasks . . . . .	10
2.2.2	Text Representation Methods . . . . .	12
2.2.3	Word and Document Embeddings Methods . . . . .	12
2.3	Multi-label Text Classification . . . . .	14
2.3.1	Problem Transformation Methods . . . . .	15
2.3.2	Algorithm Adaption Methods . . . . .	15
2.3.3	Ensemble Approaches . . . . .	16
2.3.4	Evaluation Metrics for Classification Tasks . . . . .	16
2.4	Recommender Systems . . . . .	17
2.4.1	Recommendation Task . . . . .	17
2.4.2	Types of Recommender Systems . . . . .	18
2.4.3	Evaluation of Recommender Systems . . . . .	18
2.5	Fundamentals in Neural Networks and Deep Learning . . . . .	19
2.5.1	Deep Learning in Nutshell . . . . .	19
2.5.2	Perceptron . . . . .	20
2.5.3	Feed-forward Neural Network . . . . .	20
2.5.4	Convolutional Neural Network (CNN) . . . . .	20
2.5.5	Recurrent Neural Network (RNN) . . . . .	21
2.5.6	Activation Functions . . . . .	23
2.5.7	Regularization . . . . .	24
2.5.8	Loss Functions . . . . .	25
<b>3</b>	<b>RELATED WORK</b>	<b>27</b>
3.1	Approaches for Ontology Construction and Enrichment . . . . .	27
3.2	Approaches for Multi-label Text Classification . . . . .	29
3.2.1	Feature Selection Techniques for Text Classification . . . . .	30
3.2.2	Using Deep Learning for Text Classification . . . . .	31
3.2.3	Training-less Models for Text Classification . . . . .	32

3.3	Approaches for Document Recommendation . . . . .	33
3.3.1	Content-based and Collaborative Recommendation Systems . . .	34
3.3.2	Graph Embedding Techniques . . . . .	35
3.3.3	Graph-based Approaches for Personalized Citation Recommendation . . . . .	36
3.4	Summary and Discussion . . . . .	37
4	ONTOLOGY CONSTRUCTION AND ENRICHMENT	41
4.1	Ontology Enrichment via Word Embeddings . . . . .	41
4.1.1	Noun Phrase Extraction and Representation . . . . .	42
4.1.2	Identifying Ontological Categories . . . . .	43
4.1.3	Extraction of Semantic Relation using WordNet and Lexico-syntactic Patterns . . . . .	44
4.1.4	Ontology Enrichment Methods . . . . .	46
4.1.5	Evaluation . . . . .	47
4.2	Minimally Supervised Model for Semantic Relation Classification . . .	50
4.2.1	Methodology . . . . .	51
4.2.2	Evaluation . . . . .	52
4.3	Discussion and own Contributions . . . . .	58
5	SEMANTICALLY ENHANCED MULTI-LABEL TEXT CLASSIFICATION	61
5.1	Multi-label Datasets . . . . .	61
5.1.1	EUR-Lex dataset . . . . .	61
5.1.2	Reuters Volume Corpus I (RCV1) . . . . .	63
5.1.3	Reuters-21578 (ModApte) . . . . .	64
5.2	Semantic-based Feature Selection using Typed Dependencies . . . . .	65
5.2.1	Proposed Methodology . . . . .	65
5.2.2	Dataset and Experimental Settings . . . . .	68
5.2.3	Evaluation Results . . . . .	68
5.3	Deep Learning for Multi-label Text Classification . . . . .	72
5.3.1	Comparative Analysis Settings . . . . .	72
5.3.2	Analysis Results . . . . .	74
5.3.3	Approach Transferability . . . . .	80
5.3.4	Summary . . . . .	81
5.4	Ontology-based Training-less Multi-label Text Classification . . . . .	82
5.4.1	Proposed Methodology . . . . .	82
5.4.2	Evaluation Results . . . . .	87
5.4.3	Comparative Analysis . . . . .	92
5.5	Discussion and own Contributions . . . . .	92
6	COMBINING HETEROGENEOUS INFORMATION SOURCES FOR PERSONALIZED CITATION RECOMMENDATION	95
6.1	Personalized Citation Recommendation using an Ensemble Model of DSSM and Bibliographic Information . . . . .	96
6.1.1	Ontology Construction . . . . .	96

6.1.2	Query-based Recommendation Module . . . . .	97
6.1.3	Graph-Based Ranking Module . . . . .	100
6.1.4	Fusion Module . . . . .	101
6.1.5	Dataset and Evaluation Results . . . . .	103
6.1.6	Q-DSSM Structure Optimization . . . . .	103
6.1.7	Personalized vs. Non-Personalized Recommendation . . . . .	104
6.1.8	Comparison with other Personalized Citation Recommendation Systems . . . . .	104
6.2	Using Adversarially Regularized Graph Autoencoder for Personalized Citation Recommendation . . . . .	106
6.2.1	Problem Definition . . . . .	106
6.2.2	Methodology . . . . .	107
6.2.3	Dataset and Evaluation Settings . . . . .	110
6.2.4	Model Structure Optimization . . . . .	110
6.2.5	Comparison with other Personalized Citation Recommendation Systems . . . . .	114
6.3	Discussion and own Contributions . . . . .	115
7	SUMMARY AND OUTLOOK . . . . .	117
7.1	Summary of Thesis . . . . .	117
7.1.1	Contributions . . . . .	117
7.2	Outlook . . . . .	119
	BIBLIOGRAPHY . . . . .	123
A	APPENDIX . . . . .	139
A.1	Part-of-speech Tags . . . . .	139
A.2	List of Acronyms . . . . .	141
A.3	Supervised Student Theses . . . . .	143
B	AUTHOR'S PUBLICATIONS . . . . .	145
C	CURRICULUM VITÆ . . . . .	149
D	ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG . . . . .	151



## INTRODUCTION

---

### 1.1 MOTIVATION

Information overload is a real phenomenon in academia as a result of the tremendous growth in the number of scientific publications available on the web. Information overload describes the situation where our access to these sources of knowledge and information goes far beyond our capacity to perceive them. According to a study from the University of Ottawa, more than 50 million scholarly articles were published between 1665 and 2009 [80]. Approximately 3 million articles are published each year in scholarly journals [81]. This is a tendency that shows no sign of slowing down. The number of active peer-reviewed journals increased from 28,100 in 2014 to 33,100 in 2018 [81]. Besides, we need to consider the increasing number of predatory journals that publish a high volume of poor-quality research articles. Correspondingly, the ability to find relevant and high-quality resources, side by side, to understand the meaning of the information conveyed by these research publications is becoming progressively essential. Another example in higher education are Open Educational Resources (OER). OER are digitized learning materials, including individual resources, i.e., an article as well as complete courses that are published under an open license for students, researchers, and educators as a part of a larger trend towards openness in higher education [41]. These learning materials are mostly unstructured and can be found in different formats, i.e., full courses, papers, videos, etc. The number of initiatives and projects supporting OER is proliferating. Accordingly, the number of available learning materials and repositories is growing fast. The increasing number of projects and accessible learning resources make the process of searching for relevant resources a tedious and time-consuming task. This problem triggers the need for platforms, which aggregate data sources from publishers and institutions and enrich them with a semantic description of how these resources are related. Such platforms need to offer functionalities to support the users in exploring, understanding, and finding relevant information from the broad spectrum of information sources.

*Springer Nature* has launched *SciGraph* in 2017, a linked open data platform aggregated from *Springer Nature* and their partners in scholarly domain. This initiative aims to provide researchers with access to high-quality data from reliable and trusted sources. The platform is constantly enriched with metadata from research projects, conferences, organizations, and funders, which provides a rich semantic description of how these resources are related. Also, a sophisticated ontology and a taxonomy of subjects are used to find semantically similar resources by considering their links to similar subjects. *SciGraph* is limited to resources provided by *Springer Nature* and their partners in publishing.



Relying on ontologies, for building a global linked data platform to connect information sources from other institutions and publisher, is more convenient than using only bibliographic information for the retrieval of resources. On the one side, there are many challenges to build such linked data platform: the majority of information sources are unstructured and stored in several formats, i.e., web pages, XML, EPUB, therefore rich metadata such as (affiliation, publisher, venue) is not always available, particularly in the case of OER. On the other side, existing lexical-semantic resources, i.e., *ConceptNet* can be used to connect information sources over the different providers to have a global semantic description schema.

Building such data link platforms requires three components. First, the resources are connected using ontologies. Using ontologies, the different resources can be related by linking the resources to their corresponding concepts in the ontology. To improve the discoverability of resources, the used ontologies need to be maintained and updated. Ontology construction and enrichment should be automated to avoid the tedious and time consuming manual process.

Second, the concepts from the ontologies used should be assigned to resources. The task of assigning multiple not mutually exclusive concepts/labels to a document/resource is known as multi-label text classification. The process should be performed automatically to scale up with the rapidly growing number of available resources.

Third, recommender systems which provide personalized and relevant resources should be used. Recommender systems, in general, aim to reduce the burden of information overload by suggesting items of interest to a user. They imitate the human way of thinking by choosing as other like-minded people have chosen before when a decisive first-hand knowledge is absent. Accordingly, they adapt the recommendation to the user's interest when additional meta-information is available.

Automatic approaches proposed for tackling the three aforementioned tasks, using ML and NLP, are challenged by the data variety spanning from diverse data sources, contextual information around data, to structures and formats. In addition, the high dimensionality of data and feature space, as well as the lack of enough high-quality training examples, impose new challenges on designing and training ML models tackling the different tasks. The majority of these approaches employ time-consuming, hand-crafted feature engineering. They intensively rely on substantially rich prior knowledge in the form of structured models, NLP resources, and the availability of rich training datasets.

The recent advances and development in deep learning and in using lexical-semantic resources and their success on similar tasks, make them appealing to address the aforementioned challenges. Lexical-semantic resources, i.e., WordNet [112] build a semantic graph of words and their semantic relationships as well as additional information about the meaning of words (e.g., Wikipedia articles). Exploiting the rich knowledge embedded in lexical-semantic resources can help in extracting structured information from unstructured content to support the different NLP tasks.

The cutting-edge research field of deep learning has shown superb performance across many NLP tasks including semantic parsing [172], search query retrieval [142], sentence modelling and classification [87], name tagging and semantic role labelling

[31], relation extraction and classification [96, 178]. The deep learning models, used for NLP tasks, have been inspired by leveraging the distributed text representation in a reduced linear space using word embeddings. Mostly unsupervised word embeddings transform the words and their context to vectors of numerical values. These vectors are capable of capturing latent semantic and syntactic properties of words [111]. Word embeddings preserve linguistic regularities, such as word similarity, i.e., similar words to *Frog* are *Toad*, *Litoria*, *Ranas*, which are different species of frogs.

In this dissertation, we address three major tasks concerning the realization of linked data platforms. The tasks in hand are namely ontology construction and enrichment, multi-label text classification, and document recommender systems. The focus is not on how the techniques can be used to realize a linked data platform, instead, we review the state-of-the-art techniques solving these tasks, in order to highlight the current research gaps, and propose new approaches to overcome the aligned challenges.

## 1.2 RESEARCH CHALLENGES

Considering certain viewpoints on the realization of linked data platform using lexical-semantic resources, deep learning and word embeddings triggers interesting research challenges within the focus on maintaining and updating the ontologies used, the automatic linking of subjects/concepts to resources and the recommendation of relevant resources by considering the user's preferences and history. The research challenges addressed in this dissertation can be stated as follows:

**Challenge 1:** *The intensive reliance on complicated feature engineering and linguistic analysis for ontology construction and enrichment*

Many automatic and semi-automatic ontology learning systems that are based only on text have been proposed to facilitate the tedious manual process. These systems exploit a wide range of statistical and linguistic techniques to extract concepts and semantic relations between these concepts. Linguistic techniques are language-specific and depend on the characteristics of the language, while statistical techniques use statistics on the underlying corpora to extract concepts and relationships. The main shortcomings of those techniques are ontology coverage, propagation or errors, reliability, and required computational resources.

Linguistic techniques like lexico-syntactic patterns cover a small proportion of complex linguistic space, which leads to the deficiency of such approaches. In addition, lexical-semantic resources like *ConceptNet* [97] used in order to enrich ontologies with new concepts and semantic relations. Despite the high accuracy and good structure of such resources, their coverage is limited to fine-grained concepts. Using statistics on the underlying corpora, statistical techniques such as clustering can harvest many correct concepts and relations. However, the number of incorrect relations induced is high, which might dramatically affect the quality of the generated ontology by propagating the errors using the incorrect concepts and relations. Moreover, ML models designed for semantic relation extraction or classification relies on manually annotated corpora of sentences and rich set of features. Building and collecting such corpora is

labor-intensive, domain-specific, and requires the involvement of domain experts.

**Challenge 2:** *The high dimensionality of feature space and training overhead of existing approaches for multi-label text classification and the label imbalance in classification tasks.*

Multi-label classification can be accomplished through two main methodologies, namely problem transformation and algorithm adaptation. In the first methodology, a multi-label problem is transformed into one or more single-label problems. In the second methodology, single-label classifiers are adapted or extended to cope with multi-label datasets. For both strategies, text representation is an essential preprocessing step before the actual classification task is performed. Documents are represented as vectors of features, which are numerical values. Any token in the text is a potential feature. This means that the original feature space is in the size of the vocabulary. High dimensional and sparse feature vectors are challenging for classification algorithms. Moreover, multi-label datasets with millions of instances are increasingly common, which makes building and updating the classifiers extremely labor-intensive and time-consuming. Adding new labels might require re-training of the whole model. In addition, the performance of traditional text classifiers is influenced by the label imbalance. This means that classifiers tend to perform better for more frequent labels.

**Challenge 3:** *Providing personalized citation recommendation using multi-source heterogeneous information* In the scope of this dissertation, we study the problem of citation recommendation, where a researcher is provided with a list of relevant research publications corresponding to a query text. The substantial increase in the total number of available scientific publications on the web keeps researchers overwhelmed. That makes the process of finding relevant resources that fit the user’s preferences, more challenging and tedious, especially for junior researchers and students. Citation recommender systems mitigate this problem by suggesting a list of relevant papers that can be used as references for an author given a manuscript or parts of a manuscript. In order to personalize the recommendations, additional knowledge about the author, to whom a recommendation is to be made, should be considered. The potential information sources can be grouped into content information and citation information. Content information includes the content of publications, i.e., the title, keywords, abstract, and the full manuscript. Citation information consists of the authors and their collaborations, the publication year, the affiliation, the venue of the publication, as well as the previously cited papers by the author. Existing approaches poorly consider the latent similarities between resources concerning their citation and content information. The challenge here is how to collaborate these heterogeneous information sources to provide more personalized citation recommendation.

### 1.3 RESEARCH GOALS AND CONTRIBUTIONS

In this thesis, we investigate how the rich knowledge embedded in lexical-semantic resources and unstructured documents, as well as deep learning techniques can be used to address the aforementioned challenges of current techniques in three tasks,

namely ontology construction and enrichment, text classification and citation recommendation. In the following, we present the research goals and our contributions. This objective is divided into the following primary research goals.

**Research Goal 1:** *Designing minimally supervised methods for ontology construction and enrichment*

In the Scope of  $RG_1$ , we want to investigate how deep learning and distributed representations can be used to avoid handcrafted feature engineering in previous techniques for ontology learning. We present Onto.KOM: a minimally supervised ontology learning system that uses word-pairs with their corresponding semantic relationship as sole input in addition to existing lexical databases. It minimizes the dependence on handcrafted features and supervised linguistic modules. Onto.KOM relies on a CNN that automatically learns features from word-pair concatenation in the vector space. We study the effectiveness of using the proposed method for relation classification regarding different influencing aspects namely, the input representation, the structure of the used deep learning model and the types of semantic relations.

**Research Goal 2:** *Designing semantically enhanced models for multi-label text classification*  
In the scope of  $RG_2$ , three research goals are covered:

- **Research Goal 2.1:** *Designing semantic-based feature selection methods for text classification.*

In  $RG_{2.1}$ , we want to investigate whether leveraging the syntactic and semantic dependencies between words can improve the quality of selected features as input to text classifiers. Our contribution is a novel approach incorporating the text semantics in feature selection using typed dependencies [3].

- **Research Goal 2.2:** *Analysing the feasibility of using deep learning for multi-label text classification.*

In  $RG_{2.2}$ , we want to explore how the advent of deep learning can be used to address the main challenges in multi-label text classification under the context of small datasets of long documents, where classical techniques tend to perform better. We propose a novel approach incorporating the text semantics in the input representation to reduce the high dimensionality of the feature space. Also, we analyse how the performance of such an approach is compared to state-of-the-art techniques.

- **Research Goal 2.3:** *Designing an ontology-based training-less classifier.*

In  $RG_{2.3}$ , we want to depart from traditional approaches for text classification with intensive feature engineering and linguistic analysis. We introduce a novel ontology-based training-less multi-label text classifier. We transform the classification task into graph matching problem to have a training-less classifier which provides fair prediction accuracy for both frequent and less frequent labels [4, 5].

**Research Goal 3:** *Combining heterogeneous information sources for personalized citation recommendation*

In the scope of  $RG_3$ , we investigate how we can combine the different information sources to provide personalized citation recommender systems. We propose two approaches for citation recommendation. The first approach comprised of a query-based recommendation module and a graph-based ranking module. Next, we propose another model that leverages Adversarial Regularized Graph Autoencoders (ARGAs) to integrate bibliographic network structure (which represent the citation graph) and content information into a unified framework, and encodes them in a low dimensional, compact and continuous feature space. Based on the proposed model, we can obtain graph embeddings of the heterogeneous bibliographic network. Using the graph embeddings in combination with the history information of previously cited papers, we can provide personalized citation recommendation.

#### 1.4 STRUCTURE OF THE THESIS

Following this brief introduction, we present additional background knowledge necessary to understand further parts of this work in Chapter 2. We discuss and classify related work relevant to our contributions in Chapter 3. In Chapter 4, we introduce Onto.KOM, our system for ontology construction and enrichment from unstructured text. Chapter 5 presents our approaches for addressing the main challenges in multi-label text classification. Firstly, it presents a new semantic-based feature selection method using typed dependencies between words. Secondly, it presents an in-depth analytical study of using deep learning for multi-label text classification in the context of a small dataset of long documents. Finally, it introduces our approach of using ontologies as a base for designing a training-less text classifier. In Chapter 6, we introduce two approaches for collaborating heterogeneous information sources in personalized citation recommendation.

The thesis is concluded in Chapter 7 with an overview of the core contributions. Finally, we provide an outlook on potential future work.

This chapter presents fundamental terms and concepts which are necessary to understand the following chapters. First, techniques for ontology construction and enrichment are discussed in Section 2.1. Fundamentals for text processing and representation are discussed in Section 2.2. Machine Learning (ML) approaches for multi-label text classification and evaluation metrics are discussed in Section 2.3. An overview of recommender systems and the corresponding evaluation metrics is presented in Section 2.4. Finally, Section 2.5 gives a short overview of the deep learning structures used in our proposed approaches to address our research challenges.

## 2.1 ONTOLOGY LEARNING AND LEXICAL-SEMANTIC RESOURCES

Borst [13] defined an ontology as “a formal specification of a shared conceptualization”. Shared conceptualization imposes that ontologies should serve as a shared view of a domain’s knowledge, whereas formal means it should be represented in a machine-readable format. Ontologies play an essential role in fostering interoperability between knowledge-based systems on the semantic web, which is reflected by the reliance on a large population of high-quality domain ontologies.

### 2.1.1 *Tasks of Ontology Learning*

Ontology learning refers to the task of automatic or semi-automatic construction of the different constructive components of an ontology from natural language text [15]. Figure 1 illustrates the different aspects concerning ontology learning from text. In full automatic acquisition of ontologies, the process is handled by the system without the user’s intervention, while in semi-automatic acquisition, the user’s involvement is required.

The input for ontology learning can be fully structured, semi-structured, or unstructured text. Structured information sources like knowledge bases, database schemas, and existing ontologies represent a high-quality source of information. Using structured information, the ontology learning system extracts parts of the ontology to form a base that can be extended using statistical and linguistic techniques. However, structured information sources have low coverage of concepts and relationships. For this reason, ontology learning systems based on semi-structured information sources have been proposed. Semi-structured information sources are composed of some structured information and additional text content. Examples of semi-structured information sources are XML documents and RDF. Furthermore, ontology learning systems that use unstructured information sources have been proposed since unstructured data

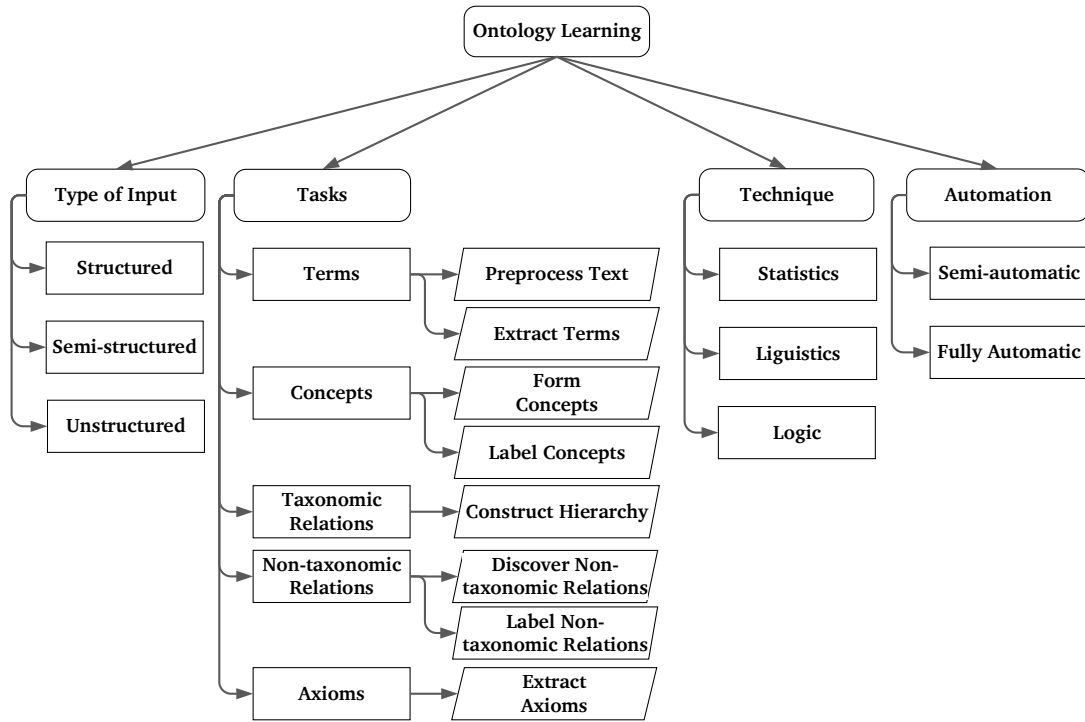


Figure 1: Aspects of ontology learning.

forms the majority of available data on the web [84]. Examples of unstructured data sources are web pages and PDF-documents.

Ontology learning procedure includes extracting terms describing the domain, forming of concepts, identifying semantic relationships between the concepts, and discovering axioms between concepts. Buitelaar et al. [15] distinguish between five different tasks in ontology learning from text.

- **Domain Terminology Extraction:** The first task in ontology learning is the extraction of domain terminology from the input text. In this step, domain-specific terms are extracted. Terms can be constituted of one or more words. Term extraction includes advanced linguistic and syntactic analysis of the text to extract complex Noun Phrases (NPs) which might express terms as well as to identify their semantic structure [15].
- **Concept Discovery:** Concepts represent an abstraction of words. After extracting the terms, concepts can be formed by grouping one or more terms with similar meanings. They can represent a natural object like an animal, an artificial object like a computer, or an abstract idea like happiness.
- **Taxonomic Relation Extraction:** Taxonomic relations correspond to sub- or supersumption relationship, i.e., “Y is-a X” and “Y is a subclass of X” between concepts. In this step, the extracted concepts are arranged to form a hierarchy of concepts.

- **Non-taxonomic Relation Extraction:** Other relationships between the concepts rather than sub- or supersumption are defined as non-taxonomic relationships between the concepts, e.g., “works-for” and “part of”.
- **Axioms Discovery:** Axioms are statements or sentences that are always taken as true [30]. They form a deduction or generalization of a set of known relationships that fulfil certain criteria. They are helpful for defining constraints, deducing other truths, and verifying correctness. For example, the subclass axiom can be verbalized as follow:
  - Each instance of *a:Child* is an instance of *a:Person*
  - *a:Child* is a subclass of *a:Person*

### 2.1.2 Lexical-Semantic Resources

Lexical-Semantic resources such as dictionaries, thesauri, and encyclopedias play a key role in many NLP problems. Dictionaries list lexical entities (a chain of words, single word, or a part of a word), and connect them based on their semantic meaning. Thesauri additionally organize lexical entities in topical groups based on their semantic relations, i.e., synonymy, hyponymy, meronymy. Encyclopedias like Wikipedia provide more detailed information of the lexical entities in the form of articles and links between the articles. In the following, we give a short overview of the different lexical-semantic resources used in our work.

#### *Wikipedia*

*Wikipedia* is a free crowdsourced encyclopedia with a large volume of high quality and comprehensive articles. The articles in *Wikipedia* provide a rich source for ontological entities (concepts and relations) through a variety of components, e.g., infoboxes, templates, categories, and internal links between articles. *Wikipedia* categories build an extensive network of links between concepts of different types. These relations can be directly projected into taxonomic relationships between concepts. Researchers have widely used *Wikipedia* as a knowledge resource for ontology learning systems [78, 86]. *DBpedia* [91] and *YAGO2* [70] are knowledge bases that have been automatically extracted from *Wikipedia* by exploiting its different constitutive components.

#### *YAGO2*

*YAGO2* (Yet Another Great Ontology [70]) is an open-source knowledge base built by extracting relations from *Wikipedia*, *WordNet*, and *GeoNames*. At the time of writing, it contains almost 17 million taxonomic relations as well as various other relation types such as “happened on date” and “lives in”. A human evaluation confirmed an accuracy of 95% of the facts in *YAGO2* [70].



## WordNet

WordNet [112] is a large lexical database. It organizes words in synonym sets (synsets). All words and phrases in a synset describe a certain concept. Furthermore, it differentiates between words in five categories: nouns, verbs, adjectives, adverbs, and function words. In addition, WordNet encodes the relation among synsets using synonymy, hyponymy, meronymy, antonymy, and morphological relations.

## ConceptNet

ConceptNet is a semantic network [97] which also provides many taxonomic and non-taxonomic relations between concepts such as “has property”, “is used for” and “located near”. Figure 2 illustrates an example of the concept “bicycle”, its related concepts and relations. It contains approximately 28 million relations and is available in 304 languages in total.

Synonyms	Related terms	bicycle is a type of...	bicycle is used for...
tr bisiklet →	en biker (n) →	en a two wheel vehicle →	en transportation →
en wheel (n) →	fr bécane →	en means of transportation →	en riding →
ja 銀輪 (n) →	en tricycle →	en a machine →	en Racing →
it bici →	en penny farthing (n) →	en ride (v) →	en personal transport →
ar دراجة هوائية (n) →	br marc'h houarn (n) →	en an efficient form of human transportation →	en ride (v) →
fr vélo →	en propel →	en toy →	en travelling on →
en cycle →	es bicicleta (n) →	en transportation →	en rush (v) →
da cykel (n) →	en like riding bicycle →	en wheeled vehicle (n) →	en cause cultural change →
it bicicletta →	ee gaso (n) →		en traveling →

Figure 2: Example of semantic relations in for the concept “bicycle” in ConceptNet [145].

## 2.2 NATURAL LANGUAGE PROCESSING

NLP is a subfield of artificial intelligence and linguistics concerned with automatic processing of unstructured (human) languages (text and speech) to represent it in machine-understandable format [32, 105]. Analysing and producing of human language is very daunting for ML and other computational approaches [54] due to the variability and high ambiguity of human languages. Words with similar meanings have no inherent relation that can be derived from the language’s structure, like talent and gift. In addition, interpreting the meaning of a text depends not only on the characters and words that form the text, but also the order and context of these elements.

### 2.2.1 Text Preprocessing Tasks

NLP encompasses various tasks exploring the semantic and syntactic meaning of words. These tasks range from low-level preprocessing tasks to complex high-level applications, e.g., text classification. Since we used different preprocessing tasks in our approaches for high-level tasks, they have to be explained shortly:

- **Parsing** Parsing is the task of analysing the structure of a text concerning its syntactic grammar [114]. Parsing identifies the syntactic relations between different elements in a sentence by forming a parse tree, such as the relation between the subject and the object of a verb [105]. Figure 3 shows an example of a parse tree.

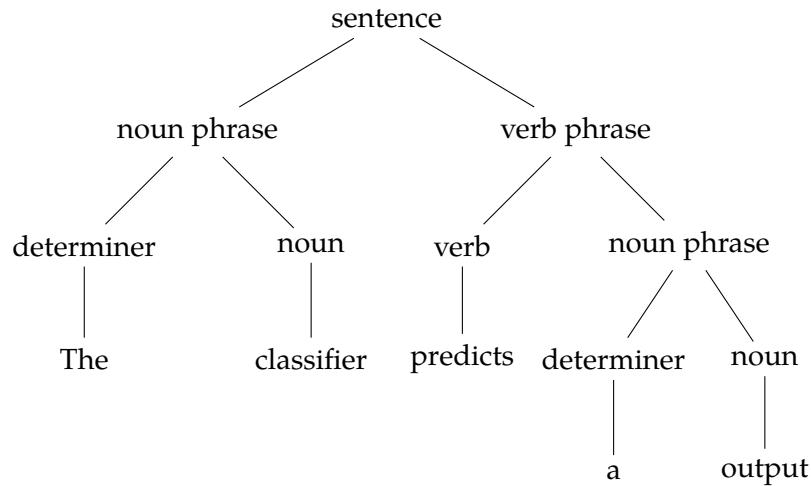


Figure 3: Parse tree.

- **Tokenization and Normalization** Tokenization is the process of converting a text into an ordered sequence of tokens. This process can be considered as a sub-task of parsing an input text. Typically a sentence is segmented by white space. The output tokens correspond to the words, numbers, and symbols in the sentence [105]. For example, the output of tokenizing “This is a way.” is the following sequence: {“This”, “is”, “a”, “way”, “.”}.

*Normalization* refers to the process of putting the tokens in their standard format. *Case folding* is a normalization method where characters are converted to lowercase. Other methods include removing punctuations, translating numbers to their word equivalents, and *lemmatization* where words are converted to their basic form.

- **Stemming** Stemming is the task of stripping off the endings of inflected words to produce the linguistic word stem form, such that different variations of a word are mapped to the same stem [114]. For instance, the stem of the words driving, drivers, and drive is driv.
- **Stop-Words Removal**

Stop-words refer to the most common words in a language that are usually removed before processing a text, such as the words “the”, “are”, and “in”. Removing stop words can help reduce the high dimensionality of feature space as they don’t contain discriminative information. Nevertheless, no fixed set of stop-words is defined for all NLP applications. Stop-Words removal is domain-specific and is sometimes inapplicable when dealing with a problem that requires exact word matching.

### 2.2.2 Text Representation Methods

Text representation is an essential preprocessing step where documents are transformed into a format consumable by ML models. This involves representing each document as a vector of words, where each dimension corresponds to the relevance of a word to the document [139]. In general, this method produces high dimensional, sparse vectors which are extremely challenging for learning algorithms. In text classification, the original feature space is the whole vocabulary in corpora because any word might be a candidate feature. To increase the manageability of the problem, ML models apply a process called dimensionality reduction which aims at reducing redundancy and noise in the data set by mapping it onto a lower-dimensional space using a wide range of feature selection and extraction techniques [156].

We can distinguish between two types of dimensionality reduction techniques, namely feature selection and feature extraction methods. In feature selection, the most representative features are selected from the original feature space based on their importance with regard to the labelset. The importance can be calculated using statistical or semantic-based techniques. Statistical methods measure the association between features and labels as a measure for the quality of a feature, i.e., Term Frequency-Inverse Document Frequency (TF-IDF), Information Gain (IG), and Document Frequency (DF). Semantic-based features account for the words order, context and similarities as a mean for calculating the importance of a feature. A simple approach is to use only NPs as features. Feature extraction methods build new optimized features, intended to be informative and non-redundant, as a function of the original feature space [156], i.e., Linear Discriminant Analysis (LDA) and autoencoders.

### 2.2.3 Word and Document Embeddings Methods

Word embedding techniques project words into a reduced dimensional space, corresponding to a dense vector of a previously specified size. Words with similar meaning or context will have similar word vectors. In our work, we consider three word embedding techniques, namely Word2Vec, GloVe, and FastText.

- **Word2Vec** is an unsupervised learning algorithm that leverages the context information to represent words in reduced feature space as a fixed size vector of float numbers [111]. Word2Vec comes in two variants, namely Continuous-Bag-of-Words (CBOW) and Skip-Gram. In CBOW the idea is given a context; we want to predict the most likely word to appear. Skip-gram is very similar to CBOW, except that the input and output are swapped. Thus, the goal is given the middle word we need to predict the surrounding words. Word2Vec is capable of capturing the context and semantic regularities of a word. This means that words with similar contexts will have similar word vectors.
- **Global Vector for Word Representation (GloVe)**: Similarly, GloVe is also an unsupervised algorithm [126]. The word embeddings are generated using word co-occurrence matrix. This matrix shows how often one word occurs in the con-

text of another word. Once the co-occurrence matrix is calculated, word vectors are learned so that their dot product equals the logarithm of the probability of co-occurrence.

$$(\mathbf{W} \cdot \vec{w}_i) \bullet (\mathbf{W} \cdot \vec{w}_j) = \log C_{ij}$$

Where  $\mathbf{W}$  is the learned projection matrix,  $w$  are the one-hot vectors corresponding to the words, and  $C$  is the co-occurrence matrix.

- **FastText** is an extension of Word2Vec. By representing words as n-grams of characters, FastText is capable of learning character n-gram representations of words in order to generate a representation of the word itself [11]. Using character n-grams, FastText trains a skip-gram model to generate the embeddings. By that, FastText is able to take subword information into account. Subword information refers to the fact that parts of the word (e.g., suffixes and prefixes) can be used to identify related words.

Document embedding techniques map documents into an informative representation as a numerical vector. Early methods for creating document embeddings relied on counting statistics on how often characters, words, or phrases occur within a document, while more recent techniques leverage the order and context of the words to create embeddings where documents covering similar topics will be close the reduced feature space.

- **Bag-of-Words (BOW):** In BOW, a document is represented as a vector of numerical values, where each dimension reflects the importance of a word. Weighting techniques like TF-IDF and Term Count (TC) are used to measure the importance of a word with regard to the document. The main downside of BOW technique is that words with the same meaning (synonyms) are considered as distinct features.
- **Bag-of-n-grams**, in which documents can be represented as an unordered collection of tokens by considering how often N-grams occur within them. By breaking long text sequences into sequences of tokens, N-grams can retain some context information compared to BOW. For example, N-grams on a character level can maintain some level of semantic information, i.e., the word “low” shares most of its trigram with its superlative “lowest”. Another advantage of character-level n-grams is reducing the feature space.
- **Doc2Vec** learns how to project a document into a low dimensional space from paragraphs with variable length. It can represent documents in a low dimensional space while preserving its nuanced semantics [90]. Two algorithms can be used for the document embeddings, namely Distributed Memory (PV-DM) and Distributed Bag Of Words (DBOW). Given a paragraph, the idea behind PV-DM is to randomly select a sequence of the paragraph and try to predict the center

word by using the selected sequence of words and the paragraph id as input. While in DBOW, by taking the paragraph id as input, the model should predict randomly sampled words from the paragraph.

### 2.3 MULTI-LABEL TEXT CLASSIFICATION

The problem of text classification is one of the fundamental tasks in NLP and Information Retrieval (IR) with many real-world applications such as spam filtering, sentence tagging, multimedia recommendation, and assigning diagnosis codes in biomedicine [43]. It has gained more attention due to the tremendous proliferation of information, most of which is unstructured.

Text classification is the task of assigning one or more labels or categories to a document. Given a training set of documents  $D = \{d_1, \dots, d_m\}$  and a predefined set of labels or categories  $\lambda = \{\lambda_1, \dots, \lambda_n\}$ , we can distinguish between two classification strategies, namely single-label and multi-label classification. Single-label classification associates each training instance with a unique label from the set of disjoint labels  $\lambda$ . Based on the number of disjoint labels, a classification problem can be identified as binary if  $|\lambda| = 2$ , or multi-class when  $|\lambda| > 2$ .

Multi-label text classification is the task of assigning a document into one or more, not mutually exclusive labels [58]. For example, a research paper can belong to two categories at the same time, namely science and machine learning. Similarly, a fiction book can simultaneously belong to three genres i.e., action, thriller, and crime. Table 1 shows a synthetic example of a multi-label dataset. Each instance can be associated with up to four different labels. The goal of the classifier is to find a mapping function which maximizes the objective function of predicting all labels related to an instance.

Table 1: An example of a multi-label dataset.

Instance	Label
$d_1$	$\lambda_1, \lambda_3$
$d_2$	$\lambda_2$
$d_3$	$\lambda_1, \lambda_2, \lambda_4$
$d_4$	$\lambda_3$
$d_5$	$\lambda_1, \lambda_3$
$d_6$	$\lambda_4$

The classification strategies that deal with multi-label problems fall into two groups, namely problem transformation and algorithm adaptation methods. We discuss both methodologies with example techniques in the next two sections.

Table 2: Transformed data sets produced by Binary Relevance (BR) method.

(a)		(b)		(c)		(d)	
Instance	Label	Instance	Label	Instance	Label	Instance	Label
d <sub>1</sub>	$\lambda_1$	d <sub>1</sub>	$\neg\lambda_2$	d <sub>1</sub>	$\lambda_3$	d <sub>1</sub>	$\neg\lambda_4$
d <sub>2</sub>	$\neg\lambda_1$	d <sub>2</sub>	$\lambda_2$	d <sub>2</sub>	$\neg\lambda_3$	d <sub>2</sub>	$\neg\lambda_4$
d <sub>3</sub>	$\lambda_1$	d <sub>3</sub>	$\lambda_2$	d <sub>3</sub>	$\neg\lambda_3$	d <sub>3</sub>	$\lambda_4$
d <sub>4</sub>	$\neg\lambda_1$	d <sub>4</sub>	$\neg\lambda_2$	d <sub>4</sub>	$\lambda_3$	d <sub>4</sub>	$\neg\lambda_4$
d <sub>5</sub>	$\lambda_1$	d <sub>5</sub>	$\neg\lambda_2$	d <sub>5</sub>	$\lambda_3$	d <sub>5</sub>	$\neg\lambda_4$
d <sub>6</sub>	$\neg\lambda_1$	d <sub>6</sub>	$\neg\lambda_2$	d <sub>6</sub>	$\neg\lambda_3$	d <sub>6</sub>	$\lambda_4$

### 2.3.1 Problem Transformation Methods

Problem transformation methods transform a multi-label problem into one or more single-label problems by converting multi-label datasets into multiple single-label datasets on which existing single-label algorithms can be used [76, 108].

Binary Relevance (BR) is one of the most commonly used algorithms for multi-label classification. The idea behind it is to create  $n$  separate datasets where  $n = |\lambda|$ , each dataset corresponds to one label as demonstrated in Table 2. Then, a binary classifier, e.g., Support Vector Machine (SVM), is trained for each label in the dataset. The multi-label classification output is then the union of all positively predicted labels. The main point of criticism regarding this method is its assumption of label independence [143].

Label Powerset (LP) maps each possible combination of labels in the training data into a new unique label. This mapping transforms the problem into a single-label classification task where the label predicted for a new instance is corresponding to a set of labels in the original dataset. This method needs the worst-case  $2^n$  classifiers, where  $n$  is the total number of labels in the training data. This approach has a high computational complexity for large values of  $n$ . Another problem is that “natural” multi-label learning data is often imbalanced, which would lead to a large number of labels with very few training samples [143]. Also, LP is unable to predict a combination of labels that were not present in the training data.

### 2.3.2 Algorithm Adaption Methods

In algorithm adaptation methods, single-label classifiers are adapted or extended to cope with multi-label datasets rather than transforming the data into different subsets of problems.

Multi-label K Nearest Neighbors (ML-KNN) is an extension of the K-Nearest-Neighbor (KNN) lazy learning algorithm using a *Bayesian* approach in order to deal with multi-label classification problems [180]. ML-KNN searches for the  $k$  nearest

neighbors of an input instance using KNN, then it calculates prior and posterior probabilities based on frequency counting of each label  $\lambda_i$  in the set of labels  $\lambda$  to determine the corresponding labels of an instance.

Multi-label decision tree (ML-DT) [28] adapts the popular decision tree algorithm C4.5 [131] in order to handle multi-label data. The C4.5 algorithm builds the classification model in the form of a tree structure. It identifies optimum breakpoints within the features based on the entropy and the information gain. It separates the dataset into smaller groups in which the values of the labels are as homogeneous as possible.

### 2.3.3 Ensemble Approaches

The term ensemble approaches refers to a group of algorithms that combine multiple ML methods [132, 183]. Random k Labelsets (RAKEL) [160] is an ensemble classifier consisting of several LPs. These LPs are iteratively constructed, each trained on a small random subset of the set of labels. For prediction, each classifier provides its decision, and RAKEL computes the average decision throughout the whole system. If the averaged score is higher than a given threshold, this label is predicted. The authors hypothesize that given enough iterations and a small labelset RAKEL is able to model label correlations, which cannot be done by using LP on its own.

### 2.3.4 Evaluation Metrics for Classification Tasks

In single-label classification *accuracy*, *precision*, *recall* and *F-measure* are the most common evaluation criteria.

For a number of classifier predictions, we have the number of *true positive (TP)*, *false positive (FP)*, *true negative (TN)* and *false negative (FN)* predictions respectively. From those numbers we can calculate the evaluation metrics mentioned below:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{F1-Measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

However, multi-label classification corresponds to predicting a set of labels, for instance. That means the prediction can be *fully incorrect*, *fully correct* or *partially correct*. A multi-label classifier can be evaluated by measuring the performance of each label separately and then averaging the results. Such schemes are called *label-based*. Another approach is to consider the average difference between the expected and the predicted sets of labels overall test examples. These metrics are called *example-based* [159].

The label-based evaluation measures assess the performance for each label separately and then the average over all labels. A micro-averaged metric  $M_{\text{micro}}$  is defined as:

$$M_{\text{micro}} = M\left(\sum_{j=1}^q \text{TP}_j, \sum_{j=1}^q \text{FP}_j, \sum_{j=1}^q \text{TN}_j, \sum_{j=1}^q \text{FN}_j\right) \quad (4)$$

While macro-averaged metric  $M_{\text{macro}}$  is defined as:

$$M_{\text{macro}} = \frac{1}{q} \sum_{j=1}^q M(\text{TP}_j, \text{FP}_j, \text{TN}_j, \text{FN}_j) \quad (5)$$

Where  $\text{TP}_j$ ,  $\text{FP}_j$ ,  $\text{TN}_j$ ,  $\text{FN}_j$  are the predictions for the  $j$ -th label. In addition, one example based metric, the Hamming Loss, is used in our evaluation:

$$\text{HammingLoss}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{\text{xor}(Y_i, Z_i)}{|L|} \quad (6)$$

$D$  is the set of examples  $(x_i, Y_i)$  with  $Y_i \subseteq L$  and  $Z_i$  is the predicted set of labels for  $x_i$ .

## 2.4 RECOMMENDER SYSTEMS

### 2.4.1 Recommendation Task

Recommender systems (or recommendation systems) suggest items to users based on the user's interest. Items are the objects to be recommended, such as books and movies. They can have multiple descriptive features, i.e., a description, category, or product image, which can be used for recommendation. The user's interest is, in general, calculated by the interactions with items. These interactions can be identified explicitly by using interactions like rate, buy and read of items by the user or implicitly based on page view, click, etc.

Given a user  $u \in U$  and an item  $d \in D$ , the recommendation algorithm tries to predict the relevance  $R(u, d)$  of an item  $d$  for a user  $u$ , using the information given on the user profile, the history of user interactions and the item profile. Based on the predicted relevance of all items for a user  $R^{\wedge}(u, d_1), \dots, R^{\wedge}(u, d_N)$ , the recommender system recommends a ranked list of items  $d_{j1}, \dots, d_{jK}$  ( $K \leq N$ ) for user  $u$ , where  $K$  is the number of recommended items [134].



### 2.4.2 *Types of Recommender Systems*

A large number of fundamental works can be found in the area of recommender systems [42]. Those works can be grouped into two main categories [134], namely Collaborative Filtering (CF) and Content-based Filtering (CBF). CF recommender systems capture the interactions between users and items to identify some implicit patterns that can be used to recommend new items. It recommends items that other users with similar interests interacted with [68, 133, 141].

CBF recommender systems build a model or profile of the user's interest based on the user's interactions with the items. It is also possible that the user profile can be constructed manually. The items are represented as a set of features. The recommendation process is based on matching the user profile against the features representing items. Also, knowledge-based recommenders are used in order to give the recommender system better reasoning ability about why an item meets the user needs [16]. Knowledge-based recommender systems require a domain knowledge that maps how a particular item meets a specific user need.

### 2.4.3 *Evaluation of Recommender Systems*

In order to evaluate the performance of recommender systems three different techniques can be followed, namely online evaluation, offline evaluation, and survey conduction.

#### *Online Evaluation*

In online evaluation, part of the users are served by a recommender system A and another part is served by a recommender system B. The recommender system, which has better performance based on some metrics (i.e., Click-Through-Rate, complexity, latency, User-Acceptance) is used [61]. This technique is considered as both costly and risky. Creating online testing systems requires much effort, time, and resources.

#### *Offline Evaluation*

Offline evaluation can be as a quantitative simulation for the online evaluation to avoid risks and high costs. Offline evaluations are reproducible and easier. Here different algorithms are used to provide the recommendations based on a part of the user's interaction. The recommendations generated are compared with the interactions in the user's history, which are not used for training the models. If an item recommended and was used by the user beforehand, it is assumed to be relevant.

For a system that needs to provide a ranked list of recommendations, the following performance metrics are used:

- Mean Average Precision (MAP): MAP is the ratio of the average precision to the number of queries. The average precision for a query is the mean of the precision obtained after each relevant item is retrieved.

- Mean Reciprocal Rank (MRR): MRR is concerned only with the rank of the first relevant item, averaged over all query texts. It is defined as follow:

$$\text{MRR} = \frac{1}{|T_P|} \sum_{i=1}^{|T_P|} \frac{1}{\text{rank}_i} \quad (7)$$

where  $\text{rank}_i$  is the rank of the highest ranking relevant item for the  $i$ th query.

- Recall@N is defined as the percentage of relevant items that appear in the top-N recommendations, where N is the number of top-N items recommended.

### *Survey*

The offline evaluation requires the availability of the user's history. A survey can be used in the case where no information is available about the users and items interactions. In this technique, synthetic sets of user with history are created then the system is asked to provide the recommendations for each user. To evaluate the recommendations, human experts are asked to provide a score for each recommendation regarding the user history [61].

## 2.5 FUNDAMENTALS IN NEURAL NETWORKS AND DEEP LEARNING

This section provides an overview on deep learning basics, including CNNs, RNNs and their structures. Then, it discusses the different hyperparameters to optimize a deep neural network.

### 2.5.1 *Deep Learning in Nutshell*

Goodfellow et al. [56] define three waves of development in neural networks. The first wave, called cybernetics, was between the 1940s and 1960s. It was intended to learn how the brain functions. The earliest models were very simple, they take a set of  $n$  inputs  $x_1, \dots, x_n$  and associate them with an output  $y$ . The model learns weights  $w_1, \dots, w_n$  associated with the inputs. These weights reflect the importance of each input.

The second wave occurred between the 1980s and 1990s. It was called connectionism. The assumption behind this wave is that using a large number of simple and interconnected computational units can achieve intelligent behavior. This is a valid assumption for neurons in the brain and for hidden units in neural networks. This wave coined the concept of distributed representations. It means that each input (i.e., word) is represented by many features, and each feature is involved in the representation of any input from the same space. Another concept introduced in this wave is the use of the back-propagation algorithm to train neural networks.

The last wave that coined the term *Deep Learning* started in 2006. The term deep learning emphasizes that it is now possible to train deeper networks than ever before. Recently, deep learning has yielded promising results over different NLP problems,

including semantic parsing [172], search query retrieval [142], sentence modelling and classification [87], name tagging and semantic role labelling [31], relation extraction and classification [96, 178].

### 2.5.2 *Perceptron*

A Perceptron is a binary (linear) classifier of single-layer neural network and the basis for more advanced neural networks. A perceptron consists of three components, namely the activation function, weight and bias, and output:

- **Weight and bias:** Each input has an associated weight, which shows the strength of a particular input. All the inputs are multiplied with their weights. Afterward, the weighted inputs are summed up, other operators (e.g., max) can be used too. The bias value is used to shift the activation function.
- **Activation function:** As activation function, a non-linear function is applied on the weighted input and passed on as output.
- **Output:** The output can serve as a prediction, or it can be passed on to one or more additional neurons for further computations.

### 2.5.3 *Feed-forward Neural Network*

A feed-forward neural network also known as multilayer perceptron, is a fully connected network. All neurons in a layer are connected to each neuron in the next layer. The structure consists of multiple layers: an input layer that provides data to the network, one or more hidden layers, and an output layer representing the network's prediction(s). Activation functions control how the information is propagated from layer to layer. A deep structure refers to a network with multiple hidden layers, where the number of hidden layers represents its depth. By using multiple hidden layers, the model can learn useful intermediate representations of the input data. Thus, an NN can learn more complex tasks, especially when non-linear activation functions are used.

### 2.5.4 *Convolutional Neural Network (CNN)*

CNNs belong to a class of deep neural networks called *space invariant artificial neural networks (SIANN)*. CNNs are regularized multilayer perceptrons that perform a convolution instead of matrix multiplication in at least one of their layers [56]. A CNN can capture the local aspects that are most informative to the prediction task. They identify local features in large structures, and combine them to produce a fixed-size vector representation of the structure [54].

A CNN consists of at least one convolutional or pooling layer and a fully connected layer.

- **Convolutional Layer:** As the name suggests, is the layer on which the convolutional operations are performed. During the convolution, a filter (Convolutional kernel) with a predefined size moves over the input vector, convolve the values within the window-size to a scalar.
- **Pooling Layer:** Pooling or subsampling operations reduce the dimensionality of the data by combining the outputs of multiple neurons into a single neuron. By that, they make the representations invariant to small transitions of the input. This can be a beneficial property, especially when the presence or absence of a feature is more important than its actual position in the data [56].

Pooling may compute a max or an average. The *max-pooling* takes the maximum value within a given window. The assumption here is that max-pooling can identify the essential information over all window positions. Also, average-pooling can be used to take the average of all values in a window instead of the maximum. Instead of performing just one static pooling operation on the output of the convolutional layer, it is also possible to retain some positional information using *dynamic pooling*. For that, the neurons are split into distinct groups, and the pooling operation is performed on each group individually. The resulting vectors from the pooling operations are then concatenated [54].

### 2.5.5 Recurrent Neural Network (RNN)

Feed-forward neural networks assume all inputs to be independent. RNNs can process sequences of input and use their internal state to learn from the sequential information. This characteristic makes them in particular interesting for NLP tasks. The term recurrent refers to the fact that the same operations are performed on every element of a given input, while at the same time, computations are influenced by the result of the previous steps. The performed computations in RNNs differ depending on the used cell. The two cells, Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU), used in our work, are explained in the following.

#### *Long Short Term Memory (LSTM)*

LSTM, proposed by Hochreiter and Schmidhuber [69], is one of the most successful cells used in RNN models. To address the vanishing gradient problem in RNNs, LSTMs use self-loops to produce paths where the gradient can flow for long durations [56]. These self-loops are weighted and controlled by three gates: an input gate, an output gate, and a forget gate. These gates control which and to what degree information is added to the cell state. The LSTM architecture halves the state vector into a memory cell and the working memory. The memory cells are designed to preserve the memory and error gradients across time and are controlled by the gates mentioned above.

Vanishing and exploding gradient are problems that can occur during the training of neural networks. The vanishing gradient occurs when the gradient backpropagating through each and every path becomes close to zero, which results in not training the network. When the gradient of the loss becomes exceedingly significant (exploding),

the learning of the network becomes unstable. While it is still an ongoing research problem, several solutions exist, including batch-normalization or the using cell architectures such as LSTMs or GRUs that assist in the gradient flow [54].

#### *Gated Recurrent Unit (GRU)*

GRU [24] was proposed by Cho et al. in 2014 to simplify LSTMs and make them less computationally expensive. In contrast to LSTMs, GRUs are only controlled by two gates: a reset gate and an update gate. Because every GRU unit has separate reset and update gates, it learns to capture different dependencies over different time scales. A unit capturing short-term dependencies will have highly active reset gates, while a unit mostly capturing long-term dependencies will have more active update gates [24].

The aforementioned RNN cells can be arranged into different models for improved results depending on the task at hand.

- **Sequence-to-Sequence (Seq2Seq)** models are often used for machine translation and question-and-answer systems (e.g., chatbots). The model consists of 3 parts: *encoder*, *encoder vector*, and *decoder*. The encoder is a stack of several RNN unites that processes the input fed into the network. The *encoder vector* is the final hidden state produced from the encoder. After processing the input by the encoder, the final hidden state is passed on to the decoder, which uses the final hidden state of the encoder as its own initial hidden state [24].

At the first timestep, the decoder gets a special token  $\langle \text{Start} \rangle$  as input to start the prediction. After this, the decoder starts the prediction of the output sequence, which can be of variable length. For every timestep, it uses the output from the previous timestep as input. The prediction ends, depending on the implementation when another special token  $\langle \text{End} \rangle$  is predicted or a predefined sequence length is reached. One particular strong suit of Seq2Seq models is its ability to predict sequences of variable length and that the length of the input and output sequence can be independent of each other.

- **Bidirectional RNN (BiRNN)** was proposed by Schuster and Paliwal [138]. The idea behind this is rather straightforward. While RNNs allow us to consider previous inputs when computing the current prediction, the following inputs may also be useful for the current prediction. BiRNNs combine an RNN that moves forward through time, from the beginning to the end of the sequence, with another one moving backward in time, from the end to the beginning of the sequence.

Given an input sequence  $x_{1:n}$ , the biRNN maintains two separate states  $s_i^f$  (the forward state) based on the input  $x_1, x_2, \dots, x_i$  and  $s_i^b$ , (the backward state) based on  $x_n, x_{n-1}, \dots, x_i$ . The forward RNN is fed the sequence in the correct order, while the backward RNN is fed the sequence in reverse. The state representation of the whole biRNN is composed of both states [54].

- **Stacked (Multi-Layer) RNN:** Stacked RNNs can be defined as an RNN model comprised of multiple RNN layers forming a grid. Consider  $k$  RNNs,  $\text{RNN}_1, \dots, \text{RNN}_k$ , where the  $i$ th RNN has states  $s_{1:n}^i$  and outputs  $y_{1:n}^i$ . The input for the first RNN is  $x_{1:n}$ . The input for the  $i$ th RNN is the output of the underlying RNN,  $y_{1:n}^{i-1}$ . The output of the whole structure is one of the RNN at the last layer  $y_{1:n}^k$  [54]. It was empirically observed that stacked RNNs work better in some tasks compared to shallower ones, for example, in a machine-translation task reported by Sutskever et al. [151].

### 2.5.6 Activation Functions

The activation functions in deep neural networks have a crucial impact on the performance of the training procedure. The activation function used influences how the information is propagated through the hidden units from the input to the output layer. Their non-linearity makes it possible for the neural networks to learn more complex patterns because linear functions cannot model simple functions such as the XOR-function. Also, computing the derivative of a linear  $f(x) = ax + b$  functions during the back-propagation would yield a constant  $a$ , which means that the input would not influence the adjusted weights.

In the following we present some of the currently most popular activation functions.

- **Sigmoid Function:** The sigmoid function (also known as logistic function) is a s-shaped function which transforms a value  $x$  into the range  $[0, 1]$

$$\sigma = \frac{1}{1 + e^{-x}} \quad (8)$$

The sigmoid function has been a staple in neural networks for a long time, but recently fell out of favour for other activation functions which prove to work empirically better [54].

- **Hyperbolic Tangent:** The hyperbolic tangent is an s-shaped function transforming a value  $x$  into a range  $[-1, 1]$ .

$$\tanh = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (9)$$

- **Rectified Linear Units (ReLU):** Rectified linear units (ReLU) is a simple and efficient activation function, defined as the positive part of its argument. It enables better training of deep neural networks compared to sigmoid function [53, 54]. They clip every value  $x < 0$  at 0.

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} 0 & x < 0 \\ x & \text{otherwise} \end{cases} \quad (10)$$

Rectified linear units are almost linear. Therefore, they preserve many properties that make linear models easy to optimize with gradient based techniques [56].

### 2.5.7 Regularization

Deep learning models are often large and consist of a high number of parameters this makes them vulnerable to overfitting. Overfitting describes a state when a ML algorithm performs well on the training set but does a poor job when used on previously unseen data. This happens because the model represents the training data too closely and thus generalizes poorly.

Regularization techniques describe a variety of different methods with the goal to prevent the network from overfitting.

- **Dropout Layer:** Dropout [148] is a technique used in neural networks to prevent overfitting. It causes the neural network to ignore a given percentage of randomly chosen neurons on the layer where the dropout is applied. One advantage of using this technique in comparison with other overfitting prevention methods is its low computational cost. It requires only  $O(n)$  computations per example per update to randomly decide which units should be set to zero in this training step. This makes the costs of applying dropout neglectable.

The drawback of dropout is that it reduces the capability of a model. To compensate that, the size of the model has to be increased, resulting in a larger model (e.g., a higher number of hidden units) and more training iterations. For large datasets, the benefits usually outweigh these drawbacks, but for datasets with very few examples, this can pose a problem [56].

- **L2 Regularization:** Another way to prevent over-fitting is by adding a regularization term  $R$  to the optimization objective:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} L(\Theta) + \lambda R(\Theta) \quad (11)$$

with  $L$  being the loss function over all parameter to be optimized  $\Theta$ . The parameter  $\lambda$  controls the degree of regularization which is applied and is set manually as a hyperparameter.

There are several options for the regularization term  $R$ . Among them  $L_2$  regularization is a more popular one.  $L_2$  regularization (also called ridge regression or Tikonov regularization), regularizes the model by trying to keep the  $L_2$ -Norm of weights low.

$$R_{L_2}(W) = \|W\|_2^2 = \sum_{i,j} (W_{[i,j]})^2 \quad (12)$$

Models using  $L_2$ -Regularization are severely punished for parameters with a high weight, but if the value of the weight is close to 0 the effect of  $L_2$ -regularization becomes neglectable [54]. This leads the model to decrease the value of one parameter with a high weight instead of a number of parameters with moderate or low weights.

### 2.5.8 Loss Functions

Loss functions map two vectors to a scalar value. These vectors correspond to the predicted and expected outputs. This scalar value is a measure of how well your algorithm models your data. By using the loss function, the gradient can be computed in order to update the weights of the network.

- **Categorical Cross Entropy Loss:** Categorical cross entropy loss (also referred to as negative log likelihood) compares the distribution of the true label  $y$  and the distribution of the predicted label  $\hat{y}$  [54]. The probability of a true label is set to 1 and 0 for other labels.

$$L_{\text{cross-entropy}}(\hat{y}, y) = - \sum_i y_{[i]} \log(\hat{y}_{[i]}) \quad (13)$$

With  $y = y_1, \dots, y_n$  being the vector representation of the true distribution over the labels and  $\hat{y} = \hat{y}_{[1]}, \dots, \hat{y}_{[n]}$  being the classifier's output by applying a softmax function corresponding to the probability distribution of the labels.

- **Binary Cross Entropy Loss:** Binary Cross Entropy Loss is used for multi-label classification problems with conditional probability outputs. The output of the classifier is transformed using the sigmoid function  $\sigma(x) = \frac{1}{1+e^{-x}}$  into the range  $[0, 1]$ , it is interpreted as conditional probability  $\hat{y}$ . The network is trained to maximize the log conditional probability  $P(y = 1|x)$  for each training sample  $(x, y)$  [54].

$$L_{\text{logistic}}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \quad (14)$$





## RELATED WORK

---

In this chapter, we review the state-of-the-art, which is related to our research and motivate our research goals. First, concerning  $RG_1$ , in Section 3.1, we discuss the related work for ontology construction and enrichment. Second, following  $RG_2$ , in Section 3.2 we analyse related work for feature selection in multi-label text classification. Then, we discuss the state-of-the-art in using the advent of deep learning for text classification in the context of a small dataset of long documents. Finally, we present the related work in using training-less models for multi-label text classification. Third, Section 3.3, discusses the state-of-the-art in personalized citation recommendation and relate them to  $RG_3$ . We conclude the chapter by discussing the challenges underlying the investigated tasks and subsequently investigating the corresponding research gaps.

### 3.1 APPROACHES FOR ONTOLOGY CONSTRUCTION AND ENRICHMENT

Automated acquisition of an ontology from unstructured text gained high attention as a result of the exponential increase in unstructured data. We can distinguish between linguistic and statistical techniques for ontology learning.

Statistical techniques rely only on statistics of the underlying corpora like word frequency, importance, and word co-occurrence, i.e., Co-occurrence Analysis, Clustering, and Term Subsumption. Co-occurrence Analysis is used to extract terms and concepts, and to discover implicit semantic relations between terms based on their co-occurrences within sentences or documents [38]. Many techniques can be used to measure the association strength between the terms like rank correlation and log-likelihood ratio [63]. In Clustering, terms are grouped using metrics for semantic similarity or semantic relatedness. The process can be bottom-up: Starting from all the terms together and dividing them in every step, or it can be top-down: Starting from single terms and grouping them based on some similarity in every step [48]. Term Subsumption is used to identify taxonomic relations between terms based on the conditional probabilities of the occurrence of the terms in documents [49].

Linguistic techniques are language-specific and depend on the characteristics of the language. Semantic Lexicons, which are digital dictionaries of words, provide easy access to a big collection of predefined concepts and relations [140]. Concepts from semantic lexicons are organized in sets of similar words (synonyms). These synonyms are employed to discover the variations of terms to form concepts. They can either be general, such as WordNet, or domain-specific, such as the Unified Medical Language System (UMLS) [95]. Syntactic Structure Analysis/Dependency Analysis is used to discover potential terms and relationships by analysing words and modifiers in syntactic structures. For instance, ADJ-NN\* or NN\* helps in the extraction of the potential terms

in the form of NPs that are most likely to be concepts in a domain. Lexico-Syntactic Patterns and Semantic Templates are employed to identify semantic relationships, i.e., hypernyms and meronyms [64]. The purpose of both techniques are quite similar but the later consists of quite detailed rules and conditions for extracting not only taxonomic relations but also complex non-taxonomic relations. Using substring inclusion and Hearst-like lexico-syntactic patterns can form an effective approach to obtain taxonomic relations [124]. Substring inclusion is based on substring matching, which can result in precise relations, for example, “bioengineering” is an “engineering”. Despite that, using Hearst-like lexico-syntactic patterns can identify relatively accurate relations; ambiguous patterns like “is a” might compromise the ontology accuracy. In addition, such patterns cover a small proportion of complex linguistic space.

Lexical-semantic resources also employed for ontology construction as they provide a rich source for ontological entities. Free crowdsourced content in Wikipedia is used for taxonomy learning and enrichment by using RDF properties and infobox triples [86, 152]. For example, taxonomic relations can be extracted using the property “rdfs:subClass” and class-instance relation can be identified using the “property rdf:type”. Maitra and Das [101] use the multilingual semantic network BabelNet [116] to establish a domain-specific hierarchy of concepts by searching for possible taxonomic relations from this semantic network. Tan et al. [153] use the endocentricity property to examine whether a given hyponym is a construction of the hypernym and some other word, for example, “goldfish” is a “fish”. This approach requires a list of word-pairs as input and uses the endocentricity property to check for taxonomic relations.

More recently, researchers used the distributed representation of words for concept and relation extraction. Word embeddings represent words in a reduced linear space where words with similar meanings will have similar representations. These distributed representations are one of the key breakthroughs for using deep learning in different NLP problems. Pombeni [125] demonstrates that word embeddings can be quite effective for extracting ontological categories. Using t-Distributed Stochastic Neighbor Embedding (t-SNE) technique to visualize the embeddings in a 2D-dimensional space, they show that different ontological categories will be relatively separated from each other in the vector space. However, the simple semantic similarity between the word embeddings is not suitable to specify the type of semantic relation between words.

Many approaches have been proposed for taxonomy construction using the distributed representation of words [137]. Cleuziou and Moreno [29] use Word2Vec vectors to calculate the cosine similarity between words. Then, they create a pretopological space to define a preliminary structure of the taxonomy. The structure is then refined by using a genetic algorithm as learning strategy to optimize the quality of the taxonomy structure and the added relationships. Fu et al. [50] use the subtraction of word embeddings to obtain a relation vector. Given a hypernym-hyponym pair ( $x$  -  $y$ ), the assumption is that there is a matrix  $\phi$  such that  $y = \phi x$ . After clustering the relation vectors, a separate  $\phi$  is defined per cluster by minimizing the L2 error over the corresponding training samples in the cluster. Now for a new word pair ( $x_{new}$  -

$y_{new}$ ), the relation vector is extracted by subtracting the corresponding word vectors of  $x_{new}$  and  $y_{new}$ . Using the relation vector, the closest cluster is selected and the corresponding matrix  $\phi$  is used to transform the vector  $x_{new}$ . If the Euclidean distance between the resulting vector and  $y_{new}$  is less than a threshold  $\delta$ , the word-pairs are predicted to have a hypernym-hyponym relation. In addition, the transitive property of hypernym-hyponym relations is used, for example ( $x$  is-a  $y$ ), ( $y$  is-a  $z$ ) implies ( $x$  is-a  $z$ ).

Ontology enrichment techniques are used in order to extend an existing ontology with additional instances and relationships. Dependency analysis, an unsupervised technique, is used to find new relations between terms by analysing the dependency information from parsing trees [27, 82]. Also, data mining methods are used to enrich ontologies by identifying hidden patterns in knowledge bases. D'Amato et al. [33] propose a data mining method to enrich the assertional knowledge in an ontology by discovering new multi-relational association rules from existent ontological knowledge bases. In addition, ML approaches are used to train models for relation classification and extraction. Different sets of supervised and handcrafted features are used for training the models, including the word-pair location in the sentence, surrounding words, Part-of-Speech (PoS), Named Entity Recognition (NER), stemming, noun compound, prefixes, and Google n-grams [14, 65, 113, 149]. Besides, lexical databases and corpora like WordNet, ProBank, FrameNet, NomLex-Plus are used to enrich the input feature with information about the words. Then the features are feed to classifiers such as MaxEnt and SVM, which in turn, learn a mapping between the features as input and the relation type as output. In addition, deep learning structures have been used for classification and extraction of taxonomic and non-taxonomic relationships [71, 150, 163, 164]. Nguyen and Grishman [118] use CNN that learns features from sentences and minimizes the dependence on external lexical databases. For training the model, they use raw sentences marked with the positions of the two entities of interest. Other researchers used lexical, and sentence level features based on word embeddings with convolutional neural networks for relation classification [162, 178].

The common characteristic of existing research in relation classification and extraction is the intensive reliance on complicated feature engineering, linguistic analysis, and external knowledge bases to provide a rich representation to feed classifiers. While some approaches use only the position annotation of word-pairs in the sentences as a feature, they still rely on a large training sets of sentences annotated manually.

### 3.2 APPROACHES FOR MULTI-LABEL TEXT CLASSIFICATION

Text classification has become a widespread problem in NLP and IR as a result of the tremendous growth of data, most of which is unstructured. In the following, we give an overview of feature selection techniques in the area of text classification. Then we discuss the state-of-the-art methods using deep learning for multi-label text classification. In the last subsection, we present approaches using ontologies to build training-less text classifiers.

### 3.2.1 Feature Selection Techniques for Text Classification

The increasing importance of multi-label classification has led to a substantial amount of research on feature selection and extraction [40, 147]. Feature selection methods can be grouped into three main categories, namely filter, wrapper, and embedded methods [147]. The filter methods select features independently of the learning algorithm. The wrapper methods select features better suited to a given learning algorithm by following a greedy search approach. Wrapper techniques are not suitable for text classification as the feature space is extremely large, and they easily overfit. In embedded methods, the feature selection is done by observing the model during training. They are used to reduce the overfitting compared with the wrapper methods. We focus on filter approaches as they fit better for text classification compared with the other two approaches. Filter approaches can be distinguished in statistical and semantic-based feature selection techniques. Statistical methods measure the association between features and labels as a measure for the goodness of a feature [44, 119, 144, 168, 169, 171]. The more widely used techniques are: *DF*, *IG*, *Gain Ratio (GR)*, *Correlation Coefficients* and  $\chi^2$  *Statistic* [170].

DF is the number of documents in which a term is present. Features are selected by defining a threshold for the DF of a term.

IG of a term  $t$  measures the level of impurity present in the information when grouping documents based on the presence or absence of this term. Let  $\{c_i\}_{i=1}^m$  denotes the labelset in the target space. IG of a term  $t$  is defined as shown in Equation 15:

$$\begin{aligned} G(t) = & - \sum_{i=1}^m P_r(c_i) \log P_r(c_i) \\ & + P_r(t) + \sum_{i=1}^m P_r(c_i|t) \log P_r(c_i|t) \\ & + P_r(\bar{t}) + \sum_{i=1}^m P_r(c_i|\bar{t}) \log P_r(c_i|\bar{t}) \end{aligned} \quad (15)$$

GR is another measure of the association between a term and a label. The estimated information gain ratio between a term  $t$  and a label  $c$  is defined as in Equation 16. Where  $N$  is the number of documents, and  $A$  is the co-occurrences of  $t$  and  $c$ .  $B$  is the occurrences of  $t$  without  $c$ , and  $C$  is the occurrences of  $c$  without  $t$ , respectively.

$$G(t, c) \approx \log \frac{A \times N}{(A + C) \times (A + B)} \quad (16)$$

Correlation Coefficients measure the strength of the relationship between a term and a label. One commonly used correlation coefficient is Pearson's correlation, as defined in Equation 17. Where  $R(i)$  is the correlation coefficient of feature  $X_i$  measured by calculating covariance and variance against label  $Y$ .

$$R(i) = \frac{\text{cov}(X_i, Y)}{\sqrt{\text{var}(X_i) \text{var}(Y)}} \quad (17)$$

$\chi^2$  Statistic measures the lack of independence between a term  $t$  and a label  $c$ . Terms are ranked following Equation 18.  $N$  is the number of documents,  $A$  is the

co-occurrences of  $t$  and  $c$ ,  $B$  is the occurrences of  $c$  without  $t$ ,  $D$  is the number of times neither  $c$  nor  $t$  occurs.

$$X^2(t, c) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \quad (18)$$

Statistical techniques ignore the word order, context, similarity, and dependencies. Considering these aforementioned aspects can lead to better representative features. One simple approach is to select the words based on their PoS tags, namely nouns, verbs, adjectives, and adverbs [93, 104]. Masuyama and Nakagawa [104] show that a much smaller feature set of nouns can perform better than combinations of other PoS tags. Fürnkranz [51] and Liu et al. [99] show that leveraging context information of word by using bi-gram and tri-gram on surrounding words can improve the performance of text classifiers. Khan et al. [85] use frequent sequence (MSF) for extracting associated sentences and co-occurring terms to improve the features selected. Also, they use *WordNet* as domain ontology to convert terms to concepts. Accordingly, they update the feature weights fed to an SVM classifier. Other researchers incorporate the ontological knowledge for training-less ontology-based text classification or to provide meta-information for feature selection [22, 26, 77].

### 3.2.2 Using Deep Learning for Text Classification

The performance of traditional problem transformation and algorithm adaptation for multi-label text classification is heavily affected by the dimensionality reduction techniques used, the preprocessing pipeline, and the feature set size. Also, the used feature selection techniques have a high impact on the performance of the used text classifiers. In the following, we discuss deep learning structures that address these challenges. Deep learning models for NLP tasks have been inspired by leveraging the distributed word representation in a reduced linear space using word embeddings, i.e., *Word2Vec* and *Fattext* (see Section 2.2.2). Word and document embeddings can form a universal representation that can be used to optimize the feature space in non-neural models. Besides, deep learning-based classifiers are capable of capturing local and global semantic features to model the correlation between labels and establish the mapping between the meaning of a label and a document.

Kim [87] uses a CNN to classify sentences. The CNN architecture is a slight variation of the one used by Collobert et al. [31]. As input, the model uses concatenated word vectors corresponding to the input words. Words that are used as input but not represented in the *Word2Vec* model are initialized randomly. Liu et al. [98] propose an extension (called *XML-CNN*) to Kim [87], which takes multi-label co-occurrences into account. This extension uses a dynamic max-pooling scheme instead of a static one, binary cross-entropy loss over a sigmoid output, and an additional bottleneck layer between pooling and output layer. Chen et al. [23] use a combination of CNN and RNN for multi-label classification. The task is divided into two subtasks: Feature extraction by a CNN and multi-label classification by an RNN. The input text is first tokenized and padded to a fixed length. Then the RNN is used for label sequence

prediction. LSTM cells are used for the RNN model. The label sequences are inserted into the neural network via an additional embedding layer. The CNN output is fed into the network as an additional linear transformation.

Berger [7] compares the performance of using CNN and RNN for multi-label classification. Additionally, a BR model with a BOW model is used as the baseline. The dataset used for the comparison is taken from the BioASQ Challenge for Large-Scale Biomedical Semantic Indexing from 2014 [158], which contains about 4.5 million short biomedical abstracts. Using the most frequent 1000 labels out of 27,000 different labels, he shows that both structures outperformed the baseline. Lenc and Král [92] propose an ensemble model of a CNN and an Feed-forward Neural Network (FDNN). As input for the network, BOW representation of the documents is used. To determine labels at prediction time, a threshold is used. The input is a sequence of words represented by their index in a dictionary with a fixed-length. Longer documents are shortened, and shorter ones are padded. Relying on BOW as input to the deep neural network will limit its performance by the quality of the input features.

Zhang et al. [182] combine non-linear embeddings using a deep neural network with modeling the label space using graph priors. During the training, high dimensional feature and label vectors are mapped into a shared embedding space. At the prediction stage, a KNN based classifier is used for the final prediction. The training phase can be separated into a label space embedding phase and a feature embedding phase. During the label space embedding phase, a label adjacency matrix  $M$  is built from the label graph structure. The label graph is structured as follows: each node denotes a label, and an edge between two labels indicates that these two labels co-occur at least for one document in the training set. Afterward, DeepWalk [127] is used to learn low-dimensional representations for the labels. After both labels and features are mapped into the same vector, one can start the predictions. This is performed by a KNN to find similar samples from the training set. The average of KNN' labels is used as the final prediction.

More recently, attention mechanisms have been used to let the model gives different attention to words/sentences for document representation. Using attention mechanisms showed improved performance on this downstream task [20, 39, 75, 173, 174, 176].

### 3.2.3 *Training-less Models for Text Classification*

Few researchers tried to use ontologies for document and label representation and mechanisms which calculate the similarity between these representations as classifiers to partially address the label imbalance and training overhead challenges. With training overhead, we refer to the time and effort required for selecting and training a ML model as well as collecting labeled training data. Zhou and El-Gohary [186] propose a domain-specific approach to classify construction regulatory documents. The labels, used for classification, form a hierarchy of topics. The topic hierarchy is manually constructed in two steps. First, environmental regularity documents are reviewed to identify the main concepts in this domain. Then, the identified concepts

are manually organized in a hierarchy following a combination of a bottom-up and a top-down approach. Similar steps are followed to build a distinct ontology for each topic. The topic ontology consists of concepts and their semantic relationships. The hierarchical softmax skip-gram algorithm is used to learn distributed representation of the terms and concepts in the corpus. Given a document to classify, the classifier computes the semantic similarity between the terms in a clause of the document and a topic from the topic hierarchy using their distributed representations. For each clause-topic pair, a Total Similarity (TS) score is computed by accumulating the similarity scores between each term in the clause and each concept in the topic ontology. Topics with positive TS are selected as potential labels.

Janik and Kochut [77] propose an ontology-based training-less classifier. In their work in the first step, a domain ontology is built using an RDF ontology extracted from the full English Wikipedia. They assume that the domain ontology has a rich instance base of interconnected entities. Given a document to be classified, the algorithm starts with converting the document to a semantic graph. The semantic graph is built by matching the document phrases with the entities in the ontology. The matched phrases are weighted based on their frequency. In the next step, the dominant thematic graph is selected by using the *hubs and authorities* algorithm. The algorithm select entities based on their importance. The importance can be computed by considering the number of associated entities. This thematic representation removes the weak and unrelated entities from the semantic graph and cleans all the noisy information by additional filtering mechanisms. Then, the most central entities in the thematic graph are defined using the shortest path algorithm. These entities are considered as categories.

### 3.3 APPROACHES FOR DOCUMENT RECOMMENDATION

In the scope of this thesis, we discuss the problem of citation recommendation as a particular case of document recommendation. The problem of citation recommendation describes the situation where a researcher (author) is provided with a list of relevant research publications corresponding to a query text. The rapidly increasing number of accessible scientific publications makes researchers overwhelmed. Citation recommender systems mitigate this problem and reduce the information overload in academia by providing papers relevant to the author's needs based on his previous publications. By relevant, we mean that the recommendation process should consider the preferences of an individual author.

Citation recommendation systems can be local or global. Local citation recommendation systems recommend a list of papers according to a short query corresponding to a given context, e.g., a paragraph for which a citation is sought. It usually ranks papers by measuring the relevance of candidate papers content to the citation context [9, 45, 74]. On the contrary, global citation recommendation suggests a list of papers that can be used as references for a comprehensive query, which usually corresponds to the entire manuscript. We mainly focus on global citation recommendations in this paper.



A naive approach for citation recommendation is to recommend prestigious authors, papers, or venues. However, this might lead to narrowing the search towards specific interest groups. Instead, such an approach, different types of information about a publication can be used for the task. On the one hand, this is the content of the publication, i.e., the title, keywords, abstract, and the full manuscript itself. On the other hand, information about the paper can also be used. Information about the paper hereinafter referred to as citation information, including the authors and their collaborations, the venue of the publication, the publication year, as well as the papers cited in the paper and their respective authors.

Personalized citation recommendation systems are characterized by the fact that they additionally use knowledge about the author to whom a recommendation is to be made. In concrete terms, these are the author's previous publications and the citations used therein. They bridge the semantic gap between the citation context (i.e. topic and authors of a query) and the cited papers by considering the citation and the content information. By this, a system can take into account, for example, the fact that an author is particularly interested in the publications of a specific venue or a particular colleague and they are interested in selected research areas only.

In the following, we will first discuss the related work in content-based and collaborative-based recommender systems. Then, graph embedding techniques will be presented. Finally, we review the state-of-the-art approaches for personalized citation recommendation.

### 3.3.1 *Content-based and Collaborative Recommendation Systems*

Early techniques for citation recommendations can be grouped as collaborative-based or content-based. CF approaches use rating matrices, generated from paper citation information [21, 83, 106, 167]. One downside of collaborative filtering is the sparsity problem. It describes the situation when available ratings are not sufficient to find similarities between users. Another problem is the cold-start problem. This refers to the fact that no recommendation can be provided for new users. Accordingly, new papers will not be recommended until having enough ratings. Moreover, the lack of transparency in collaborative filtering systems yields a recommendation without reasoning. CBF approaches [66, 83, 115, 179] are limited by the quality of the representative features and whether the representation captures all the aspects that might influence the author's preferences or interests. To mitigate the disadvantages of both strategies, hybrid recommender systems were proposed [107, 157]. However, they still inherit some of their limitations too.

Personalization in the recommendations can be realized by defining user and item profiles. Then the recommendation engine defines rules that control how items are recommended to users. In the following, we discuss relevant approaches in the area of scholarly paper and learning content recommendation. Pujahari and Padmanabhan [130] build a user profile using decision lists. The recommender learns rules of the user preferences based on the decision lists. Then, documents are ranked based on

matching against the defined rules. Rohani et al. [135] use a hierarchy of categories to classify documents. The categories are stored in the inner nodes of the hierarchy while the documents are stored in the leaves. Then, the user profile is represented as a hierarchy of preferences. The preferences correspond to the interest level in the categories. Using the user's history, the recommender assigns a score to each node. This score represents the user's interest in a category. New documents are recommended by selecting the top 10 categories with the highest scores. Then, documents similar to the documents associated with these categories are recommended.

Ontologies can be used to provide context-aware recommender systems that account for the user context (i.e., location, time, and learning goals) to provide personalized content. Yu et al. [177] propose a recommender system for context-aware E-learning. The system relies on three ontologies. First, the *Learner Ontology* which contains information about the user's learning goals, learning interests, location, and the subjects already mastered. Second, the *Learning Content Ontology* which defines the properties of contents and maps them using a set of relationships. Third, the *Domain Ontology*, which is built using existing ontologies and taxonomies. The system ranks the learning contents according to the learner's goal in the *Learner Ontology*.

Recently, researchers have proposed deep learning approaches for tackling recommendation problems [25, 62, 161, 187]. Some of these approaches rely solely on deep learning structure [46, 67, 165]. Gong and Zhang [55] propose a hash-tag recommender system that solely relies on a CNN for tags recommendation in microblogs [55]. They transferred the recommendation task into a multi-class classification problem. Xu et al. [165] present a tag-aware personalized recommender system that uses a deep semantic similarity-based structure in order to compute the similarity between a user and an item, where both the user and the item are represented by tag annotations. Similarly, other deep learning models: CNN, RNN, and Deep Semantic Similarity Model (DSSM) are used [181]. The capability of deep learning to learn the non-linear relationship between users and documents as well as to implicitly learn the representative features of document and user profiles are causing a paradigm shift towards the deep learning-based recommendation.

### 3.3.2 Graph Embedding Techniques

Network representation techniques are used to encode network structure (citation information in the context of citation recommendation) and content information into a low dimensional, compact and continuous feature space. We can distinguish between two types of graph embedding algorithms [122].

First, approaches that assume that only network structure information is available, and the learning objective is to maximize the likelihood of preserving network neighborhoods of vertices. Perozzi et al. [127] propose DeepWalk algorithm, which uses local information collected from random walks as input and uses it to learn a latent representation of the vertices using a neural network model. Grover and Leskovec [59] propose Node2Vec, which differs from DeepWalk in the sense that Node2Vec can use a Breadth-first Sampling (BFS) or a Depth-first Sampling (DFS) algorithm to search for

neighboring vertices. Tang et al. [154] develop LINE to infer low dimensional representations for each vertex by defining first-order and second-order proximity which must be preserved by the representations. The major limitation of these approaches is that they assume a homogeneous network of objects and ignore the content information.

Second, approaches that assume node content information is available and exploit both network structure information and content information simultaneously. Yang et al. [166] propose text-associated DeepWalk (TADW) which incorporates content information into network representation. Pan et al. [123] propose TriDNR approach, which uses three types of information, namely network structure, vertex content, and vertex labels, to jointly learn vertex representation.

The mentioned algorithms largely ignore the latent distribution of the embeddings [122], which may lead to poor representation when dealing with sparse and noisy data. This limitation was the motivation for adversarial models which regularize the embeddings to follow a predefined distribution. Adversarial models are controlled by a Generative Adversarial Network (GAN) [57]. GAN plays an adversarial game with two linked models: the generator  $G$  and the discriminator  $D$ . Makhzani et al. [102] develop an adversarial autoencoder. The system consists of two components: the autoencoder and the generative network. Dai et al. [34] adapt the adversarial autoencoder to be applied to graphs. Accordingly, Kipf and Welling [89] develop a Variational Graph Autoencoder (VGAE). Just like a 'normal' Graph Autoencoder (GAE), but instead of using the encoder to directly map the data to the representations, VGAE maps first to a mean and a variation which are then finally used to produce the representations. Pan et al. [122] propose an adversarially regularized graph autoencoder based on VGAE. The generator is used to regularize the generated embeddings to force the embeddings/-posterior distribution of the autoencoder to follow a normal distribution  $N(0, 1)$  which is called prior. The resulting architecture has again a structure-preserving module and a generative adversarial network. But in this case, the structure-preserving module is an autoencoder.

### 3.3.3 *Graph-based Approaches for Personalized Citation Recommendation*

Graph-based techniques have been exploited for citation recommendation by transforming the recommendation problem into link prediction. The flexibility to incorporate different citation and content information in the graph gave graph-based approaches more attention.

Ohta et al. [120] use a set of technical terms extracted from the user's previously browsed papers to retrieve a list of related papers. Then they build a bipartite graph consisting of the related papers retrieved and the technical terms appearing in these papers. The top-ranked papers are recommended based on their importance, measured by analysing the bipartite graph using the Hyperlink-Induced Topic Search (HITS) algorithm. Similarly, Chakraborty et al. [19] build a bipartite graph of paper citations and keywords to provide recommendations in response to a search query. The keyword graph links papers with their keywords. The citation layer creates a directed link between papers and their citations. Next, they create clusters of keywords. Using

the generated clusters, papers containing the keywords in a particular cluster form the candidates list recommendations. To ensure a balance between prestige papers and diversity in the recommended citations, they use the Vertex Reinforced Random Walk (VRRW) algorithm. Pan et al. [121] build a multi-layer graph of citations, key terms, and their relationships. The citation layer is represented as an undirected graph. WordNet is used to compute the similarity between the key-terms in the graph. The key-terms for each paper are selected based on ranking the candidate terms using TF-IDF scores. Then, a ranked list of papers is recommended by applying a graph-based similarity learning algorithm.

Other researchers improve personalized recommendations by incorporating co-authorship and venue information [110]. Using a probabilistic neural model, Huang et al. [74] propose an approach that jointly learns a distributed semantic representation of citation context and cited papers. The recommendations for queries are provided by training a multi-layer neural network model to estimate the probability of citing a paper given a citation context. Cai et al. [17] propose a three-layered reinforced model. The first layer is an undirected graph and links papers based on their citations. The second layer is built between authors using their co-authorship relations. The third layer is the venue layer; it links venues according to their thematic similarity. The layers are interlinked by considering the authors and venue information of papers. Using a three-layered interactive clustering approach, a subgraph generated by the clusters associated with a query text is selected, and the corresponding papers are recommended.

More recently, researchers used GAE to learn latent embedding of the vertices in the bibliographic network and use it for citation recommendation [18, 184]. Zhang et al. [184] build content-based and author-based graph representations. Then, the concatenation of both representations is used as the paper feature vector. Papers are recommended based on their cosine similarity to the query text. In a similar approach, Cai et al. [18] use an Energy-based GAN [185]. The content information representation is done using a content2vec module that incorporates all the text content associated with one paper and all other papers written by the same author. The input for the model is the concatenation of content information and the adjacency matrix. The main limitation of this work is that the adjacency matrix corresponding to the cited papers is used as input. Cai et al. [18] did not state how a citation recommendation can be derived for a new query since the citations are not known.

### 3.4 SUMMARY AND DISCUSSION

We reviewed the related work with respect to the three tasks considered in this work, namely (i) ontology construction and enrichment in Section 3.1, (ii) multi-label text classification in Section 3.2 and (iii) personalized citation recommendation in Section 3.3. The main goal of this work is to propose new models that leverage the rich information embedded in lexical-semantic resources and unstructured documents as well as deep learning structures in order to overcome the challenges of the state-of-the-art models for the three aforementioned problems. Accordingly, we investigate how dis-

tributed representations in combination with deep learning structures can contribute to the different problems. In the following, we present the identified research gaps and our contributions.

Ontologies play a key role in linking the vast amount of information sources on the web. These ontologies need to be constantly maintained and updated with new concepts and relationships. Many approaches were proposed for relation classification and extraction. Their common characteristics are the intensive reliance on complicated handcrafted feature engineering, linguistic analysis and external knowledge bases to provide a rich representation as input to classifiers. Despite that, recently proposed approaches based on deep learning structures rely only on the word-pair location and context, they still need a large corpus of manually annotated sentences to train the models. Collecting and annotating the training samples is a labor-intensive and time-consuming task. The sentences should be manually annotated and collected by domain experts.

In the Scope of  $RG_1$ , we investigate how distributed representations and deep learning structures can be used to minimize the intensive reliance on complicated feature engineering and linguistic analysis for ontology construction and enrichment. In order to overcome the challenges mentioned, we introduce in Chapter 4 Onto.KOM: a minimally supervised, fully automatic and domain-independent ontology learning system. Our approach assumes that a first ontology exists, which is aligned with our assumption of how ontologies can be used for building linked-data platforms. We distinguish our work from previous research in two aspects. Firstly, by clustering the words based on the similarities between the correspondent word embeddings, we can automatically identify the different ontological categories in a text corpus. This is done in an unsupervised manner by using validity indices to select the optimal number of ontological categories. Secondly, we develop a new model that uses a CNN structure for multi-way classification of semantic relations between words by using the embedding concatenation between word-pairs as the sole input.

Using multi-label text classification techniques, resources can be automatically assigned concepts from the ontologies.  $RG_2$  focuses on analysing how the rich information embedded in the unstructured text can be leveraged to improve the feature selection process. Moreover, we study the application of knowledge bases and deep learning techniques to overcome the high dimensionality of feature space, label imbalance and training overhead. In related work, we show that researchers have incorporated text semantics in feature selection by selecting NPs or n-grams as features, whereas others tried to leverage external lexical databases, e.g., *WordNet* to enhance the performance more by using it for selecting relevant concepts. However, extracting ontological associations using external lexical resources or patterns has shortcomings due to the small coverage of concepts for particular domains and thus less ontological entities can be acquired. Previously, we have proposed methods for incorporating semantic knowledge into feature selection by selecting NPs appearing in taxonomic relations as candidate features [3]. However, the applied patterns to extract taxonomic relations might work with significantly lower precision for more natural texts. Also

the number of discovered taxonomic relations will be significantly lower. In this work, presented in Section 5.2, we extend and improve these semantic-based methods further with a new approach using typed dependencies to extract syntactic relations between NPs as a mean for measuring the importance of a NP, aiming to achieve better performance even with more natural text.

With regard to using deep learning for multi-label text classification, a massive amount of data is necessary to optimize the different hyperparameters. Also, the majority of reviewed methods assume relatively large datasets of short text blocks or fixed input length. However, in many real-world applications, we don't have a relatively large number of labeled training samples. Typically traditional classifiers tend to perform better than deep learning models when only a small amount of training data is available. Respectively, document length might vary from few words to thousands of words. That triggers the need for alternative solutions to optimize the input feature space, which consequently will help in reducing the number of hyperparameters to be optimized. This was the starting point for our investigations. In Section 5.3, we compare different methods and parameters using a dataset that does not fulfil the frequently assumed properties. The focus of our work is to analyse how common deep learning structures can be used as a basis for more complex models to minimize the supervised feature engineering and the tedious classifier selection analysis in the context of a relatively small dataset of long documents.

Label imbalance is a persistent problem in traditional problem transformation and algorithm adaptation methods as well as in deep learning-based methods for text classification. The classifiers tend to perform better for frequent labels while the performance drops for infrequent labels. The existing approaches for training-less multi-label text classification rely on manually built ontologies by domain experts, structured text and predefined categories hierarchy. These assumptions significantly limit the usability of these approaches in other domains where such ontology or domain experts are not available. In Section 5.4, we improve previous work by introducing a new ontology-based training-less classifier which assumes unstructured documents and does not rely on manually built domain-specific ontologies.

Recommender systems can eliminate information overload in academia by providing users with relevant resources based on their preferences. We take the problem of citation recommendation as a special case of document recommendation. The reviewed approaches leverage both citation and content information. Ontology-based approaches mainly consider the content of previously browsed resources. They can provide high-quality recommendations. However, these ontologies need to be steadily maintained and updated. Moreover, the user profile and the content profile assume the availability of rich metadata which limits the system scalability and applicability over different domains.

In the scope of RG<sub>3</sub>, we address the problem of collaborating the different heterogeneous information sources (citation and content information) in order to provide personalized citation recommendation. The previously studied models which use solely deep learning as a recommender system, still rely on labelled data to train a deep semantic model to learn the relationship between items and users. Collecting

such labelled data is time-consuming and tedious. In Section 6.1, we propose an automated approach to train DSSM structure to measure the semantic similarity between two text blocks. Using the proposed approach, we elaborate and improve on previous research for personalized citation recommendation by proposing an ensemble model of two modules, namely a query-based recommendation module and a graph-based ranking module.

However, in this approach, both citation and content information are separately considered. Other researchers used graph embeddings models to encoded the network structure and content information in a lower-dimensional feature space. In our approach, presented in Section 6.2, we use a GAN-based autoencoder to obtain a graph-based representation of the content and citation information. Using this approach, papers are recommended based on the weighted importance of the cosine similarity calculated using the graph embeddings and the citation history.

In this chapter, we address the first research challenge  $RG_1$  (see Section 1.2): *The intensive reliance on complicated feature engineering and linguistic analysis for ontology construction and enrichment*. Based on the identified research gaps, we investigate how deep learning structures and word embeddings can be used to overcome the aforementioned challenge in current approaches. The more recent techniques are ML-based and require high-quality training data. However, building and annotating sentences as training examples for semantic relation classifiers is a tedious and time-consuming process. Lexical-semantic resources, as presented in Section 2.1.2, form a rich source of ontological entities (mainly, concepts and semantic relations). The question that arises is: how can we leverage these resources to build and enrich ontologies with the aid of deep learning techniques?

In Section 4.1, we design and evaluate a novel system for ontology construction and enrichment from unstructured text, named Onto.KOM. In this system, we cluster the word vectors to identify ontological categories, which correspond to groups of words with similar semantic meaning. Then, given a basic ontology extracted from WordNet, we evaluate multiple approaches for enriching the ontology with new concepts and relations by using the word embeddings of word-pairs as the sole input feature.

In Section 4.2, we extend Onto.KOM to design a minimally supervised, fully automatic and domain-independent system for semantic relation classification. The goal of this extension is to perform multi-way classification of semantic relations using CNNs. In our work, we evaluate this approach and the effectiveness of using word embeddings and CNN to identify and classify new semantic relations. We focus on three main aspects: the input representation, the CNN structure and the performance variation over different types of semantic relations.

Partial results of the work presented have been previously published by the author of this thesis in [2].

#### 4.1 ONTOLOGY ENRICHMENT VIA WORD EMBEDDINGS

Our approach automatically extracts new relations from an unstructured text by relying on the word embeddings of word-pairs. First, we extract all single and multi-word terms from our corpora. The extracted terms represent the domain terminology. Then, we identify different ontological categories by clustering the words using the corresponding embeddings. We use validity indices to measure the quality of resulting clusters. The ontological categories are topical categories representing words with similar meanings. The output of the first step are the different ontological categories, i.e., science, food, and animals. Secondly, using lexico-syntactic patterns and WordNet, we build a robust sub-ontology for each ontological category. Using the sub-ontologies,



we train a separate classifier for each ontological category in order to identify new semantic relations. The constitutive components of our approach, shown in Fig. 4, will be explained in the following:

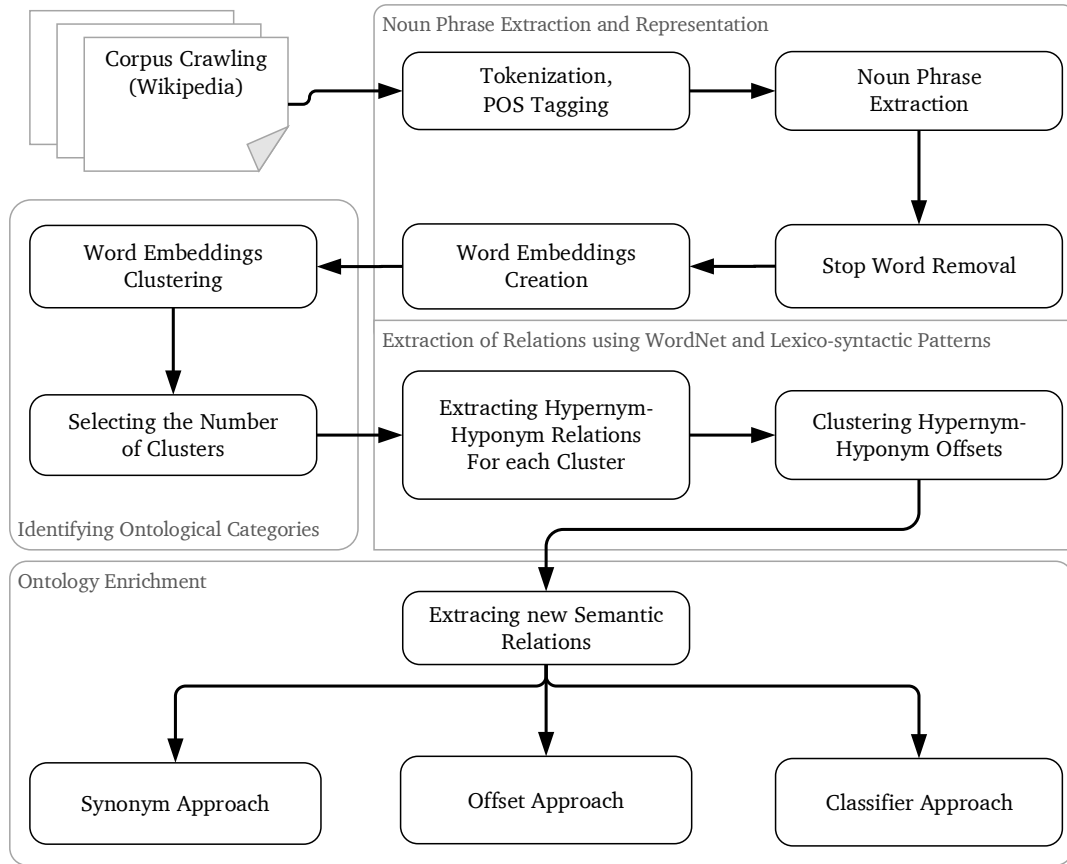


Figure 4: Block diagram of the proposed ontology learning system

#### 4.1.1 Noun Phrase Extraction and Representation

In order to extract the domain terminology from a corpus of documents, we use linguistic filters to extract all NPs. The domain terminology forms the basis for our semantic relation extraction phase. The role of the linguistic filters is to recognize essential terms that may represent concepts or instances of concepts. In addition, they filter out sequences of words that are unlikely to be concepts. The linguistic pipeline includes tokenization and PoS tagging. In PoS tagging, the words are tagged as corresponding to a particular PoS, i.e., noun, adjective, verb. A combination of three linguistic filters is used to extract multi-word NPs that can reflect essential concepts:

- Noun Noun+
- Adj Noun+

- (Adj| Noun) + Noun

In the next step, the corresponding word embeddings are generated. Creating word embeddings for multi-word terms, like *artificial intelligence* directly from text corpus will lead to losing critical information about this kind of word constructions. In order to enable the learning of these very common constructions, we concatenate all multi-word terms (e.g., *artificial intelligence*  $\rightarrow$  *artificial\_intelligence*), then we train a model to learn vector representation of concatenated terms.

We evaluate two techniques for creating the word embeddings, namely Word2Vec and GloVe. For GloVe, we use one configuration with a vector size of 300 and a minimum number of occurrences of 5, a window size 15 and 30 iterations. The parameters are set based on the experiments conducted in [126], which compare GloVe against other word embeddings models except Word2Vec. We evaluate multiple configurations for Word2Vec. We evaluate both models on two tasks, namely the semantic similarity and semantic relatedness. Jastrzebski et al. [79] combine 17 established datasets in the categories of similarity and analogy in order to evaluate word embeddings. For the final evaluation, we use six datasets, namely WordSimilarity 353, 353R, MEN, MTurk, SimLex999, and 353S to benchmark the created embeddings on similarity related tasks. Correspondingly, three datasets, BLESS, the Google analogy dataset, and SemEval2012, are chosen for the assessment of analogy related tasks. Based on the average performance on similarity and analogy tasks we decided on using GloVe embeddings in our approach.

#### 4.1.2 Identifying Ontological Categories

Word embeddings preserve linguistic regularities, such as word similarity and analogy. Figure 5 illustrates the projection of word vectors corresponding to NPs from a subset of 6274 Wikipedia articles covering the artificial intelligence category into two-dimensional space using t-SNE. Using hierarchical clustering with  $K = 20$  to cluster the 300-dimensional word vectors, we can identify relatively separated ontological categories. Concepts belonging to *machine learning* and *statistics* are adequately separated in the vector space. These results indicate a strong clustering effect, thus a good separation between words belonging to different ontological categories can be achieved.

A primary decision regarding clustering the word embeddings is which clustering method to be used and the number of clusters. Clustering validity indices have been widely used to specify the optimal number of clusters based on the quality of the clusters produced [36]. In this work, we select the optimal number of clusters based on the majority vote of three indices, namely Dunn, Davies-Bouldin, and Silhouette. Also, we use these indices to select the clustering method used. We evaluate two clustering methods, namely hierarchical clustering and K-means. The lower score of Davies-Bouldin index indicates better clusters quality while a higher score for Silhouette and Dunn indices refers to better-separated clusters.

Figure 6 illustrates the scores for Dunn and Davies-Bouldin indices over the different number of clusters. K-means has higher scores than the hierarchical clustering

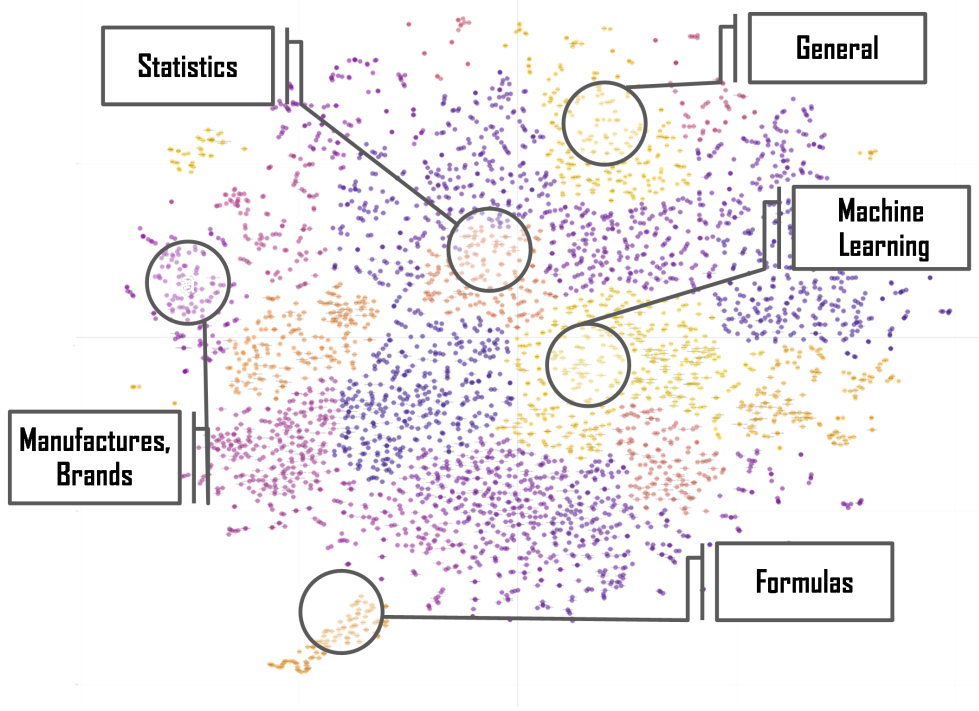


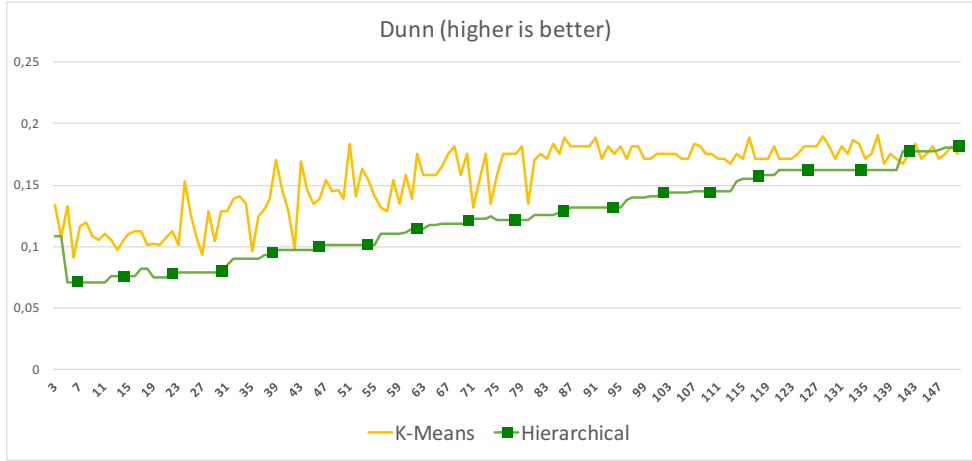
Figure 5: The distribution of word vectors from artificial intelligence articles using t-SNE plot.

approach using the Dunn index as shown in Fig. 6a. However, in the case of more than 145 clusters, hierarchical clustering outperformed K-means. From Fig. 6b, we notice that the indices for K-means highly fluctuate due to the random selection of initial centroids. In contrast, the results for hierarchical clustering show that this technique produces more stable results with a low variance in the indices scores over the different number of clusters. We proceeded using a hierarchical clustering approach based on the relative comparison of the indices' scores for both algorithms.

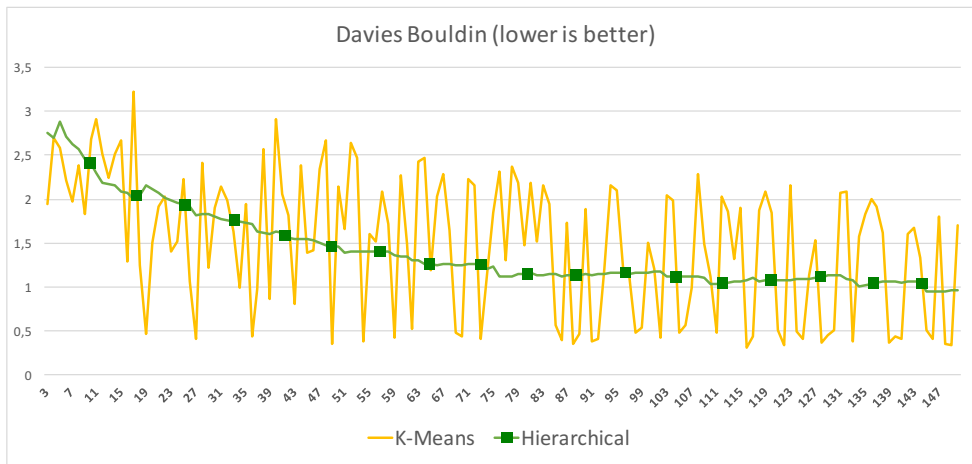
#### 4.1.3 Extraction of Semantic Relation using WordNet and Lexico-syntactic Patterns

Concepts related to different ontological categories, i.e., *food* and *animals* occur in different contexts. For that, their semantic relations have varied perspectives. Consequently, building a separate model for each category to classify the semantic relations within the different categories is an essential step to improve the system's overall performance. Therefore, for each resulting cluster, we build a robust ontology by adding semantic relations between the terms. The high quality of the ontology is essential to minimize the error propagation in the ontology enrichment phase. We want to achieve high precision of the relations and take low coverage into account. For that, we rely on lexico-syntactic patterns and external lexical databases to create this ontology. We use the Hearst six lexico-syntactic patterns:

- NP such as {NP,<sub>i</sub>} \* {(or| and)} NP



(a) Dunn index.



(b) Davies-Bouldin index.

Figure 6: Results for two validity indices in relation to the number of clusters.

- $sush\ NP$  as  $\{NP, \} * \{(or|and)\} NP$
- $NP \{, NP\} * \{, \}$  or other  $NP$
- $NP \{, NP\} * \{, \}$  and other  $NP$
- $NP \{, \}$  including  $\{NP, \} * \{(or|and)\} NP$
- $NP \{, \}$  especially  $\{NP, \} * \{(or|and)\} NP$

Currently, we use WordNet as a proof of concept to extract taxonomic relations. However, extracting ontological associations using WordNet has short-comings due to the low coverage of concepts for particular domains but this aspect is for future work.

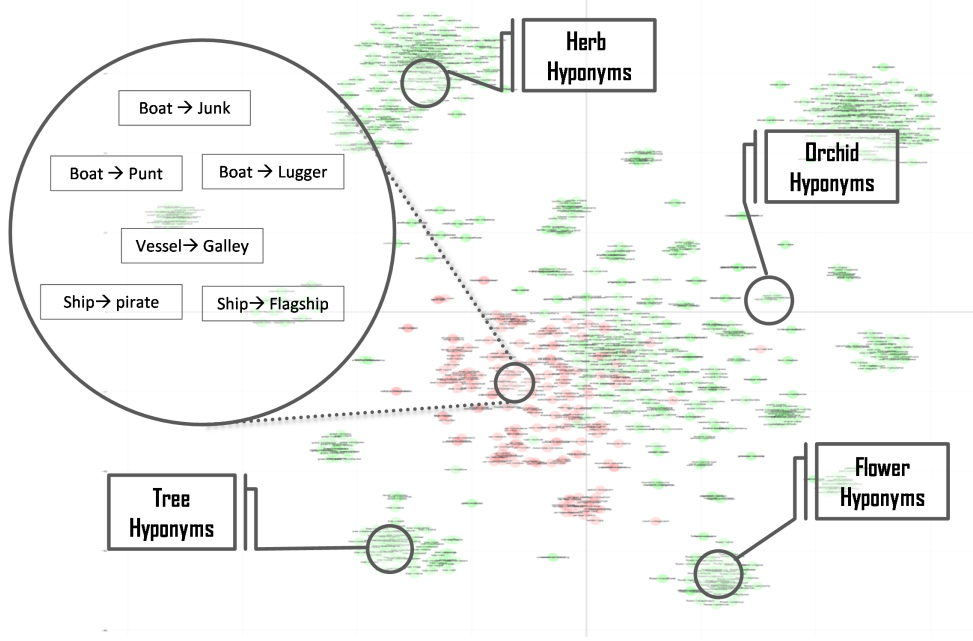


Figure 7: The distribution of the taxonomic relation offset for the plants and vehicle categories using t-SNE plot.

#### 4.1.4 Ontology Enrichment Methods

Ontology enrichment methods are used to extend the existing ontology with additional instances and relations. Figure 7 illustrates the embedding offset of hypernym-hyponym relations from concepts of two different domains, namely *plants* and *vehicles* using t-SNE. The Figure shows that relation offsets (embedding offsets) are adequately distributed in clusters, which implies indeed that, these offsets can be decomposed to form more fine-grained relations. This implies that word-pairs with the same relations will be close in the vector space and thus have the potential to be used for discovering new relations. In the following, three different methods to discover new relations, namely the synonym, offset and classifier approaches will be introduced.

##### *Synonym Approach*

The basic assumption for this approach is that for a given hypernym-hyponym relation  $(X, Y)$ , one can find new relations with the same hypernym  $X$  by searching for "synonyms" of  $Y$ . For the relation *coupe*  $\rightarrow$  *car*, searching for similar or semantically close words for *coupe* will lead to *compact*, *convertible*, *roadster* or *sedan*. In combination with the corresponding hypernym *car*, new taxonomic relations can be found. The idea in respect to word embeddings is that words similar to  $Y$  should be close in the vector space. The procedure to find an alternative for  $Y$  is to find a number of word vectors  $v_{Y'}$  that are closest to  $v_Y$  the vector representation of  $Y$ , based on some threshold  $\delta$ :

$$\text{distance}(v_Y, v_{Y'}) < \delta \quad (19)$$

While identifying many correct relations, this naive approach might also create a high number of false positives. In order to improve on this approach, for a given hypernym  $X$  and a set of hyponyms  $Y_0, Y_1, \dots, Y_N$  an alternative  $Y'$  has to be a shared alternative between at least  $n$  hyponyms in the top  $K$ -Nearest results. For example for  $n = 2$ , the hypernym-hyponyms relations *compact*  $\rightarrow$  *car* and *convertible*  $\rightarrow$  *car*, the word *roadster* has to be in the closest  $k$ -nearest for both *compact* and *convertible* to be considered as a new hyponym of *car*.

#### *Analogy Approach*

This approach uses the similarity between the offset of the hypernym-hyponym word pairs in order to find new relations. The offset between two vectors  $X, Y$  corresponds to their arithmetic difference. This approach is similar to the work of Pocostales [129], however, instead of learning offset projection, the idea is to find similar embedding offsets based on the embedding offset of all correct hypernym-hyponym relations. Similar to the synonym approach, this approach utilizes a  $k$ -nearest neighbor model with either euclidean or cosine distance as a threshold.

#### *Classifier Approach*

Enriching the ontology with additional relations based on the embedding offset is more complex than reliance on similarity scores. Moreover, the taxonomic relations in *vehicles* domain are spatially close, but separate from the relations in the *plants* domain which entails the need to create a separated model for each category. For that, we investigate the feasibility of using the embedding offset between two words as the only input to three different classifiers, namely SVM, Conditional Inference Tree (Ctree) [72] and CNN.

#### 4.1.5 *Evaluation*

The English Wikipedia was used as a corpus for creating the word vectors. We have used hierarchical clustering with *Dunn*, *Davies-bouldin* and *Silhouette* to identify the different ontological categories. In addition, we used Stanford CoreNLP toolkit [103] in this work for performing the different NLP tasks (POS, linguistic filter and taxonomic relations extraction). This toolkit combines ML and probabilistic approaches to NLP with sophisticated, deep linguistic modelling techniques.

In the first evaluation step, we analyse the feasibility of using word similarity and relatedness for ontology enrichment. Two ontological categories, namely *vehicles* and *plants* were used to evaluate the three different approaches. The initial semantic relations, extracted from WordNet for both categories, form the basic ontology. With regard to the generated word embeddings from Wikipedia, the coverage for the *plants* category was 952 relations from 4,699 in WordNet, while 208 relations from a total of 585 for *plants* were found. This shows the need for more domain-specific corpora or crawling more articles from the web.

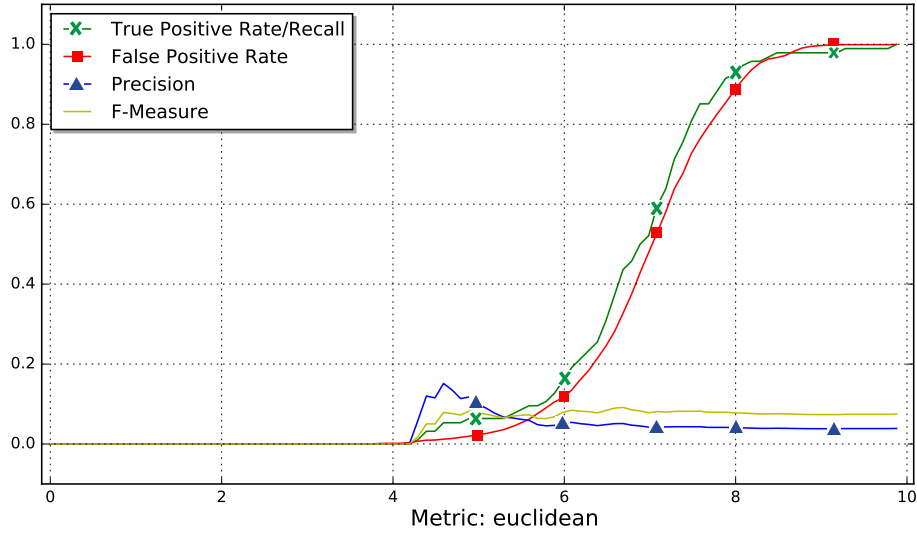


Figure 8: Results for synonym approach based on similarity threshold in *plants* domain.

Figures 8 and 9 subsequently show the associated graphs of different performance metrics with regard to the similarity threshold for both domains using euclidean distance. It is clear that the distance distribution for correct and incorrect synonym relations are similar, which indicates that using only the distance threshold to identify new relations will have poor performance.

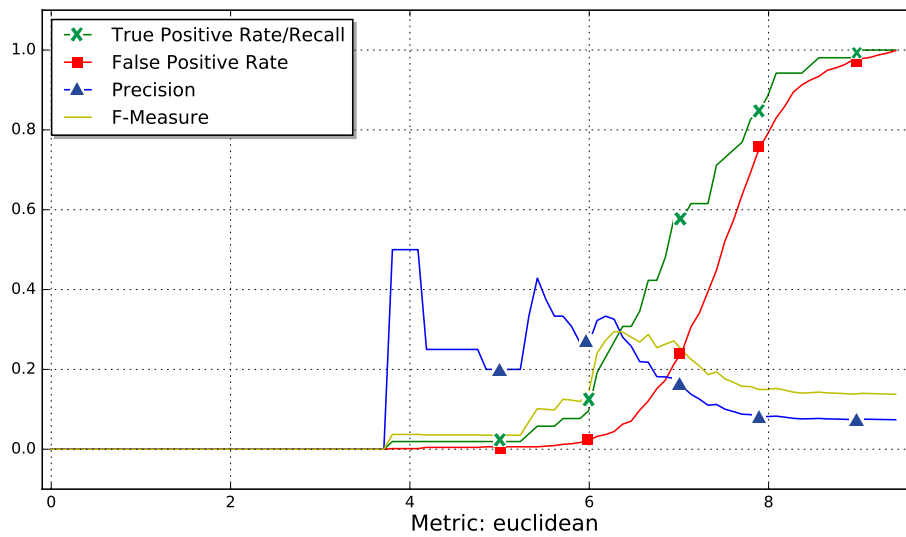


Figure 9: Results for synonym approach based on similarity threshold in *vehicles* domain.

With the offset approach, Figures 10 and 11, show a better distinction between false and true relations based on the embeddings offset. However, with small distance threshold, many correct relations will be misclassified, while with high distance threshold many false relations will be classified as correct taxonomic relations.

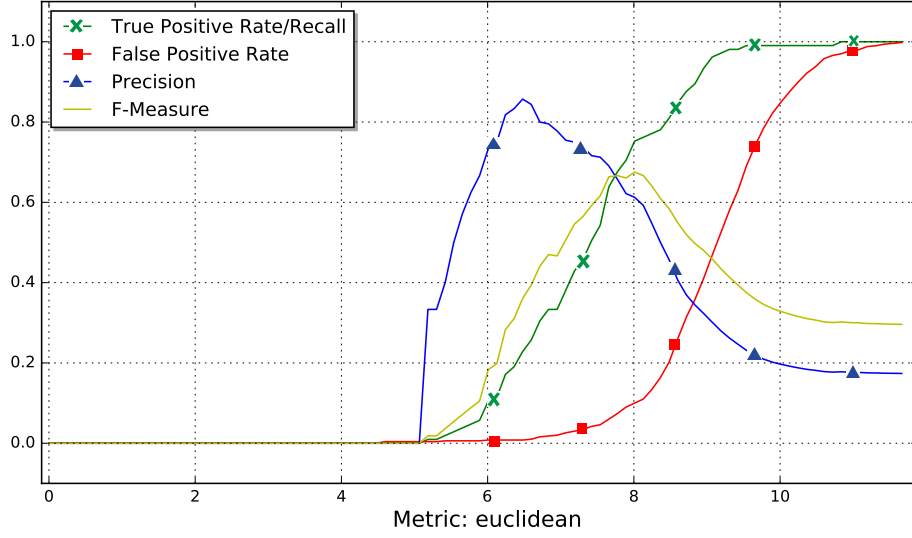


Figure 10: Results for offset approach in *plants* domain based on distance threshold.

Based on the analysis of the first two approaches we can conclude that the embedding offset is more complex than what similarity distance can imply. For that, we tried three different classifiers following different paradigms, namely SVM, Ctree and CNN. In order to train the classifier on negative examples too, a set of 1000 random relations for both domains was extracted from WordNet synsets without taxonomic relations. For the CNN network configuration, we used L2 regularization and an initial learning rate of 0.01. Each filter is initialized using *Xavier* initialization [52]. We trained our model with a batch size of 200 over 30 iterations, with *Stochastic gradient descent* as optimization algorithm and *Nesterov* [117] as an updater function with momentum of 0.9.

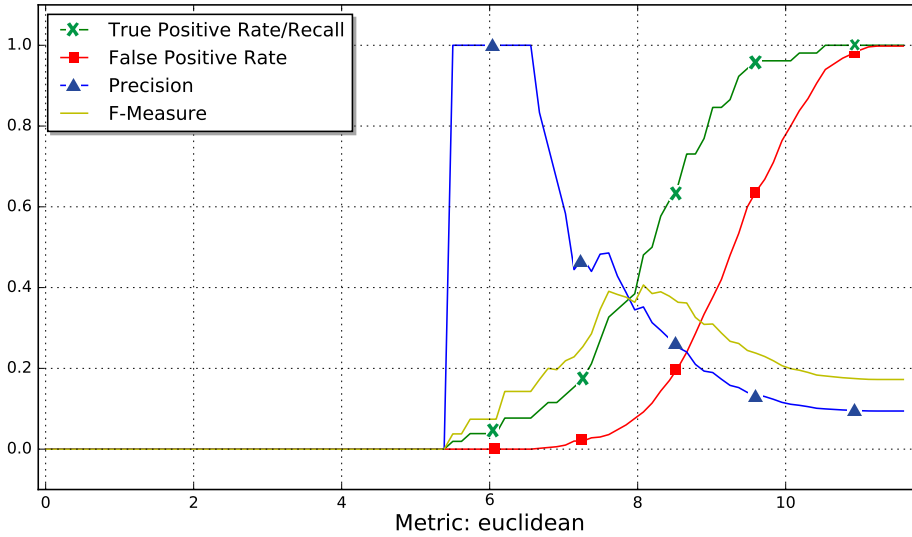


Figure 11: Results for offset approach in *vehicles* domain based on distance threshold.



Table 3 provides the results of a comparative analysis of approaches from the related work against the proposed CNN classifier as well as SVM and Ctree with the embedding offset as the only input for taxonomic relation classification over a combined dataset of both domains. This experiment presents an average performance of the designed model. The results of 5-cross validation folds are promising. The CNN model without any additional designated features is capable to provide the best performance for taxonomic relation classification and better than other classifiers with exterior features. Using the same input representation, the CNN shows better performance compared to SVM and Ctree.

Table 3: Comparison with other classifiers for taxonomic relation classification.

Approach	Features	F1-Score
SVM	PoS, syntactic patterns, stemming	55.2
SVM	word pair, words in between	60.7
SVM	embedding offset	53.2
Ctree	embedding offset	53.0
<b>Proposed CNN</b>	<b>embedding offset</b>	<b>82.7</b>

#### 4.2 MINIMALLY SUPERVISED MODEL FOR SEMANTIC RELATION CLASSIFICATION

In this section, we describe an extension of the previously presented approach to design a multi-way semantic relation classification model. Our model uses a CNN classifier for predicting different types of semantic relations between words by using the embedding concatenation between word-pairs as the sole input. In addition to *WordNet*, we use *ConceptNet* to enrich the basic ontology with more semantic relations. Due to the results of the evaluation presented in the previous section, we mainly focus on analysing the input representation, the CNN structure and the performance variation over different types of semantic relations.

First, we determine the concepts of the ontology by extracting all single and multi-word terms from a specific corpus (steps A1 - A3 in Figure 12). Secondly, we add relations between the concepts, by (B1) crawling relations between the terms in our terminology from external lexical databases and knowledge bases, namely *WordNet* and *ConceptNet*. In addition, (B2) we add relations by crawling the corpus using lexico-syntactic patterns. By means of these two steps, we build a base robust ontology. This ontology, or more precisely (C) the embedding concatenations (resp. subtraction) between the word-pairs of the relations from the ontology, are used to train one or several classifiers (D), This classifier(s) serve(s) to identify and classify semantic relations.

The proposed approach is minimally supervised because lexico-syntactic patterns and knowledge bases are used only for training the classifier without the need for manually collected and annotated sentences. Having a basic ontology with coverage

of concepts from different domains will make the system capable of implicitly identifying semantic relations between words, without frequent co-occurrences based on capturing their context similarity. For a new text corpus, the system should be capable of identifying the different semantic relations using the word vectors and without any additional feature engineering. The constitutive components of the extended version of Onto.KOM, shown in Fig. 4, will be explained in detail in the following:

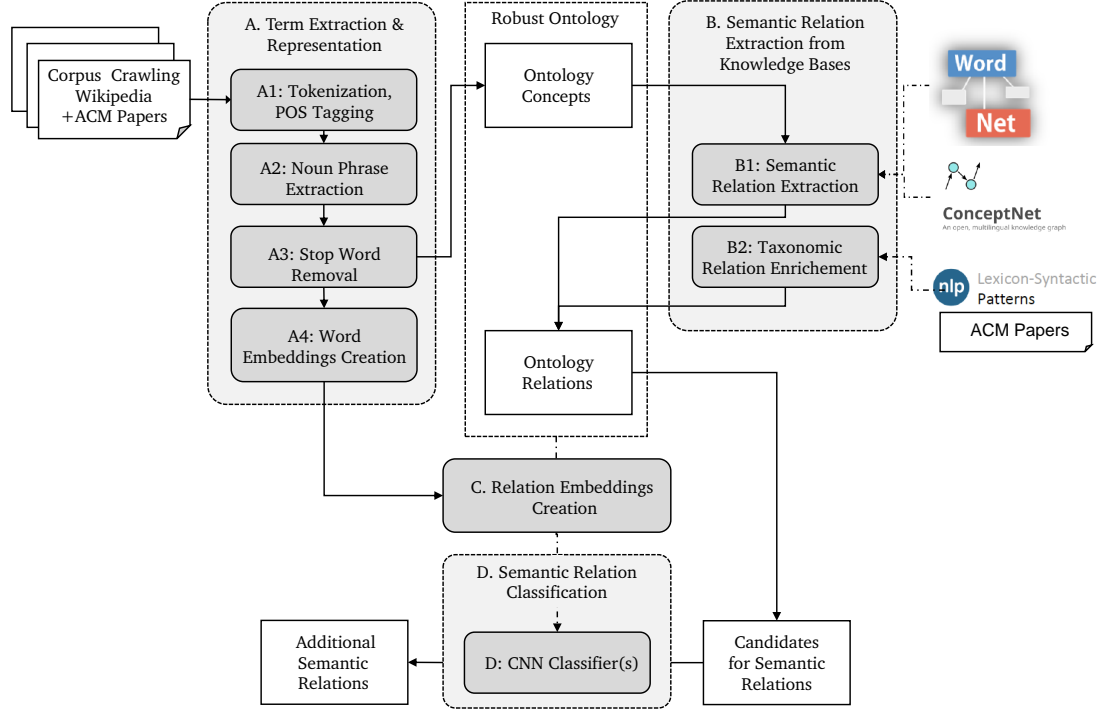


Figure 12: Block diagram of the proposed system for semantic relation classification.

#### 4.2.1 Methodology

In the first steps (A1-A3), we identify the concepts of the ontology by extracting all NPs from the corpus. NPs are extracted using the same steps in Section 4.1.1. Similarly, multi-word NPs like “*Supervised Machine Learning*” will be concatenated as “*supervised\_machine\_learning*”.

Then, (A4) the word embeddings of the extracted NPs, including the concatenated term, are created. The word embeddings are used in a later step for training the classifier. Both GloVe [126] and FastText [11] are candidates for creating the word embeddings. FastText could be especially useful when working with very specific multi-word concepts that do not occur in the preprocessed corpora as a full match. Parts of these specialized concepts could still be recognized so that a meaningful representation of the concepts can be generated.

Recently proposed contextualized word embeddings, i.e., Flair [1], BERT [37], and ELMO [128] showed improved performance on many downstream tasks. However, they're designed to incorporate the context of a word in the sentence to generate the corresponding vector, while our assumptions based on individual words or phrases and their context in a sentence isn't considered, for that Fasttext or GloVe might be better since they consider words independently.

In the next steps (B1 and B2) we build a robust base ontology by adding semantic relations of different types between the concepts of the domain ontology. Since we want to generate a high-quality ontology, we rely on external lexical databases and knowledge bases, namely *WordNet* and *ConceptNet* to find the initial relations (B1). *WordNet* contains different kinds of semantic relations between nouns, namely synonymy, hyponymy, meronymy, antonymy, and morphological relations. *ConceptNet* [146] provides many non-taxonomic relations such as "has property", "is used for" and "located near".

In order to add more taxonomic relations, we (B2) rely on the six Hearst lexico-syntactic patterns [64] extracted from the document corpus. For other semantic relations, i.e., meronomic relations other statistical or linguistic methods could be used, i.e., semantic templates and association roles, but we are not doing that at this point, It is the subject of further work.

Since the lexical databases used are static and have small coverage of concepts for particular domains, we have to add additional semantic relations. For this purpose, we train a CNN (D) to learn semantic regularities contained in the robust ontology. This CNN is used later on as classifier. The existing relations are fed to the CNN in form of relation embeddings (C). These relation embeddings are created by using the list of tuples where each tuple has the form  $(Word_1, Word_2, Relation_{id})$ , more precisely the concatenation (resp. the subtraction) of the word embeddings of the word-pairs. As we will see in Section 4.2.2 the trained CNN is capable of predicting the presence of semantic relations between unseen word-pairs. Since for training the classifier, the embeddings concatenation (resp. subtraction) between word-pair is the only feature, no manual feature engineering is necessary.

In the following, we analyse and optimize the different constitutive components of the CNN classifier. Another important consideration when handling multiple relation types is whether to use one classifier per relation type or one classifier to output all relations.

#### 4.2.2 Evaluation

The experiments aimed to study the effectiveness of using the word embeddings and a CNN network to classify and identify additional semantic relations. We first analyze three different aspects, namely the input representation, the network structure and whether to use one classifier for all relation types or several classifiers. Based on the findings of these aspects we define a system configuration that is used afterward to evaluate our approach on well-known classification tasks and compare it with other approaches.

Wikipedia and 100,000 ACM papers were used as corpora to train enrich the ontology and train the CNN classifier. The original GloVe and FastText implementations, used in step A4, are trained using the English Wikipedia and the ACM papers with an embedding vector size of 300. As noted earlier, the lexical databases *WordNet* and *ConceptNet* were used to find initial relations between the concepts in step B1. Additional relations were extracted in step B2 from the ACM papers which provides domain-specific knowledge. We kept the relations occurring at least 3 times to guarantee high precision of the extracted relations. This threshold was defined based on a manual inspection of a random sample of the retrieved relations. To generate additional relations from *Wordnet*, we have used word-pairs that do not have a direct relation by exploiting the transitive property of taxonomic relations. Relation embeddings are then created in step C using two different methods, subtraction or concatenation, whose performance is also evaluated.

For the CNN classifier, we use multiple layers of strided convolutions with ReLU activation function. In the first experiments, only one CNN is used for all relation types. The number of filters in the first layer is chosen to be 32 and is doubled in every layer until a maximum of 128 filters is reached. The convolution filters use a size of 3 with the exception of the first 2D convolution which uses a size of (7, 2). Batch normalization is used in all cases. RMSprop was used as the optimizer. Other modifications to the classifier are evaluated later. Tensorflow and Keras [60] were used to implement the CNN. Additionally, we have used L2 regularization and an initial learning rate of 0.001. Each filter is initialized using *Xavier* initialization [52]. We trained our model with a batch size of 256 over 10 epochs.

#### *Semantic Relation Representation*

We compared two ways of forming the relation embeddings, namely concatenation and subtraction, by evaluating the performance on a train and test split of four *ConceptNet* relations, namely “Derived from”, “Form of”, “Is a” and “Synonym”. FastText word vectors were used. Table 4 shows that concatenating the word vectors outperformed the offset representation on all classes of relations. We hypothesize that this is because the information is lost when using the offset representation instead of preserving the information of both words together. We used concatenation in all the following experiments.

Table 4:  $F_1$ -Scores for classification of *ConceptNet* relations using different input representations

Configuration	Derived from	Form of	Is a	Synonym	Average
Concatenate	62.2%	67.7%	83.7%	70.6%	71.3%
Subtract	56.7%	60.6%	72.3%	60.1%	62.4%

### Word Embeddings: GloVe vs. FastText

We compared the performance of GloVe and FastText embeddings using the same four semantic relations from *ConceptNet* as in the previous experiment. The words in the test set are partially included in the generated word vectors. Due to the results of the previous evaluation, the embedding concatenation of the word-pairs was used as the CNN features set. As GloVe can only get vectors for words that occurred in its training corpus we compared the performance when using only words available in the GloVe embeddings and the performance when using all words by substituting unknown words with GloVe’s unknown token for both the train and test set. Table 5 demonstrates that, while GloVe performs similarly to FastText on the known words, FastText outperforms GloVe when using all words. Therefore, FastText was used in all the following experiments.

Table 5: F<sub>1</sub>-Scores for classification of *ConceptNet* relations using different subsets of words for GloVe and FastText

Used Data	GloVe F <sub>1</sub> -Score	FastText F <sub>1</sub> -Score
GloVe words only	70.9%	71.2%
All words	68.5%	71.3%

### Different Classifier Structures

We have explored different modifications of the classifier structure and different hyperparameters using identical relations from *ConceptNet*. Table 6 summarizes the results. We evaluated the performance of different configurations using twice as many filters, noise in the input, L2 regularization, using an additional unstrided convolutional layer before every strided convolutional layer and using dropout with a 50% chance of retaining the values. We also experimented with using one classifier per relation instead of one classifier to output all of them. We found that the system using twice as many convolutional filters without any other modifications performed best. In all cases, we used the same hyperparameters which were found using manual optimization. We also tried greedy optimization which we quickly discovered to be infeasible due to the large number of possible hyperparameters. We then tried using hyperopt which gave slightly worse results than the manual optimization. For the classifier optimization itself, we tried standard SGD, Adam and RMSprop. We found that Adam and RMSprop performed similarly and finally decided to use RMSprop.

Additionally, we found that dropout does not increase the accuracy and additional regularization techniques such as L2-regularization and adding input noise even decrease it significantly. This is likely because of the huge number of relations available in this test. Later for the non-taxonomic relation classification dropout proved useful and without it, our unmodified system would quickly overfit on the small training set. It can be seen that as the number of examples shown increases so does the classification performance of the system.

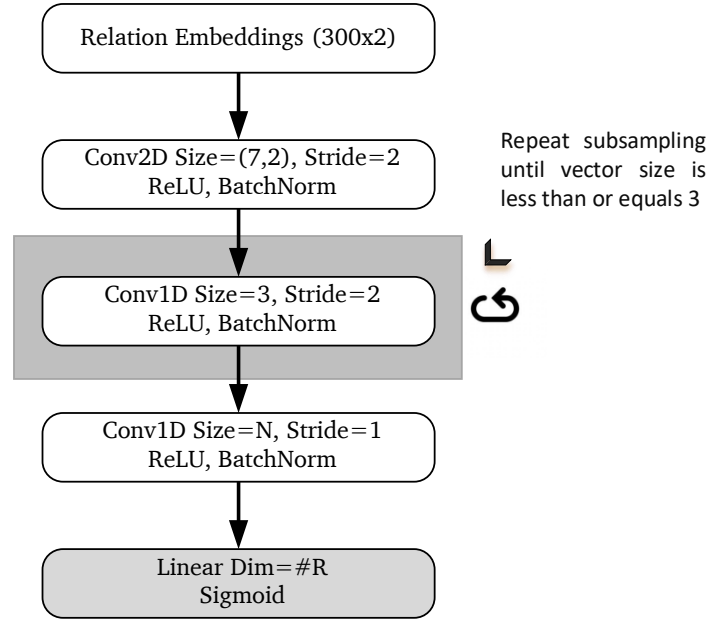


Figure 13: Structure of the final CNN classifier.

It can be seen that while having a separate classifier for each relation type yields higher performance than using a combined classifier, it also uses many more parameters. Doubling the number of filters and thus quadrupling the number of parameters in the combined classifier yields a better average  $F_1$ -Score than having separate classifiers using roughly the same number of parameters. We assume this is because the combined classifier can learn to make use of the additional data to learn better representations for the relations, whereas each separate classifier would have fewer data to be used for training. Doubling the number of convolutional layers also yielded a small improvement. Doubling both the number of filters and convolutional layers was worse than only doubling the number of filters. This could be since, without other regularization techniques, this big network overfits on the training data. What we did not evaluate here is the runtime necessary for training the classifier. While doubling the number of filters results in an approximately 4% improvement, it also quadruples the number of necessary computations. The final configuration we used in the rest of this work can be seen in Figure 13 (although a softmax layer is used at the end instead of a sigmoid layer in the case of multi-class classification).

#### *Multi-Way Classification and Training Set Size*

We have evaluated our final configuration (using FastText and concatenation) on the SemEval 2010 Task 8 competition for multi-way classification of non-taxonomic relations between word-pairs, marked in a sentence, under ten categories, e.g., “Cause-Effect (CE)” and “Instrument-Agency” [65]. For this task specifically, we also used 50% dropout after every layer as there only very few training examples exist. We evaluated

Table 6: F<sub>1</sub>-Scores for classification of *ConceptNet* relations using different modifications to the classifier structure

Configuration	Derived from	Form of	Is a	Synonym	Average
Combined,	62.2%	67.7%	83.7%	70.6%	71.3%
Combined, Dropout	65.8%	65.4%	79.0%	71.8%	70.5%
Combined, Dropout, Noise	62.4%	62.9%	81.7%	70.3%	69.3%
Combined, Dropout, L2-Reg	44.8%	61.3%	81.7%	66.0%	63.4%
Combined, 2 Convs	65.5%	<b>71.5%</b>	79.4%	75.0%	72.8%
Combined, 2x Filters, 2 Convs	66.6%	71.1%	81.6%	<b>77.4%</b>	74.2%
Separate	<b>70.4%</b>	67.3%	82.9%	73.2%	73.4%
Separate, Dropout	66.1%	65.1%	82.7%	73.3%	71.8%
Combined, 2x Filters	68.6%	71.1%	<b>83.6%</b>	77.2%	<b>75.1%</b>

the system’s performance on the four training subsets as defined in the competition (using 1000/ 2000/ 4000/ 8000 training samples). The results are shown in Table 7. A significant improvement can be seen in all metrics when using more training data. Furthermore, we experimented with using the nearest neighbour classification for test samples that had a very small cosine distance to a training sample but saw no difference to what the CNN classifier was already predicting. Additionally, we experimented with using dilated convolutions but that showed no improvement.

Our model is ranked 5<sup>th</sup> compared to the participants in the challenge without further optimization of the structure. The best performance reported on this dataset is using *Att-Input-CNN* [162], who achieved F<sub>1</sub>-Score of 87.2 which clearly outperforms the model here. However, they rely on substantially richer prior knowledge in the form of labelled sentences with the corresponding relations, while Onto.KOM relies on embedding concatenation between word-pairs as the sole input, which minimizes the reliance on rich domain-specific knowledge. For that, in the next experiment, we show that by having more word-pairs corresponding to a specific relation we can outperform complex models with intensive dependence on rich prior knowledge.

Table 7: The performance of the final model on the test set by using different training subsets of the SemEval 2010 Task 8 challenge

#Training Samples	Accuracy	Macro Precision	Macro Recall	Macro F <sub>1</sub> -Score
1000	53.4%	52.8%	54.4%	52.9%
2000	56.7%	57.3%	56.9%	56.4%
4000	62.3%	62.3%	63.3%	62.4%
8000	65.9%	65.6%	67.1%	65.8%

### *Taxonomic Relation Classification against the State-of-the-Art Approaches*

Using lexical databases it is possible to obtain a large number of taxonomic relations that can be used for training. What we then still require for training the CNN are the negative examples. Some form of creation of false relations is necessary. We started out by choosing two random words as unrelated word-pair. After training the model, we noticed poor performance. This can be attributed to the network recognizing the words that occur in the lexical databases and predicting any occurrence of them at all as a relation. Then we tried choosing only words that occurred in the lexical databases. This resulted in the network learning to predict by the position of where a word occurred. For example, if the lexical database has many relations where a word appears on the left-hand side of the “is a” relation, the classifier would always predict that a relationship where that word appears on the left-hand side is a true relation. To circumvent this we ended up generating two false relations for every true relation by taking one word from each relation and choosing a random word out of the lexical databases for the other one.

Using the word-pairs with taxonomic relations from the different combinations of the datasets described in Table. 8, we evaluate our system on AI-related trial data of SemEval 2016 Task 13 competition [12]. This set was selected since it includes 2385 word-pairs manually evaluated to be correct taxonomic relation or not. Less than 13% of these taxonomic relations are included in the used datasets and were not used to train the classifier. This assists us to evaluate the learning capability of our CNN. In this challenge, the goal was to build a taxonomy given a list of words. While the goal of our system is not to build a taxonomy we can still evaluate our accuracy on this trial data as it provides both true and false relations. We decided not to use the taxonomic relations from YAGO as they contain many relations that are not useful to us such as relations containing names of people. Table 8 shows the accuracy using a classification threshold of 0.5.

Table 8 shows that using *ConceptNet* “is a” relations provided lower performance compared to *WordNet* and ACM relations. This is because *ConceptNet* relations contains many singular and plural pairs that are not useful for our task. Using the *WordNet* hyponyms we can achieve significantly better performance. Using the extracted ACM hyponyms we obtain worse performance than the *WordNet* hyponyms even though the data seems just as good. This is because we have fewer examples as we only take the words that occurred at least 3 times.

Moreover, we compared our model against the *Taxi* method for taxonomy induction [124], that reached the first place in the *SemEval 2016* challenge on taxonomy extraction evaluation. The method extracts taxonomic relations using Lexico-Syntactic Patterns, Substrings and Focused Crawling from large general domain corpora and domain-specific corpora bootstrapped from the input vocabulary which is the AI trial in this experiment. Table 8 shows the performance against the baseline. The *Taxi* method achieved an accuracy of 24.9%, although that we considered both direct and indirect (*hypernym,hyponym*) pairs. This is because the method relies on syntactic properties of the words or explicit co-occurrence of word-pairs in sentences, around 80% of the extracted relations are substring-based (i.e., *training corpus, corpus*).



Table 8: Comparison against two approaches from related work on the AI trial of SemEval 2016 Task 13

Method	Accuracy
Taxi (all relations found)	22.1%
Taxi (with pruning)	24.9%
Att-Input-CNN	32.5%
Onto.KOM (ConceptNet)	41.4%
Onto.KOM (ACM)	57.5%
<b>Onto.KOM (WordNet)</b>	<b>68.6%</b>

In addition, we evaluated the *Att-Input-CNN* structure [162] following two scenarios. Firstly, we extracted taxonomic relations using Hearst patterns from 3760 articles under the AI category from Wikipedia in order to train the classifier. Around 11492 taxonomic relations in 7209 distinct sentences were extracted, however, only 16 word-pairs from our test set were found, thus this scenario can not be used to evaluate the model. Secondly, we crawled all sentences where the word-pairs of the AI test set co-occur. We found 10731 sentences and used them as positive examples, while we used the dataset from the challenge as negatives and divided the resulted dataset into 80% training and 20% validation. By testing different configurations, the best classifier trained with 100 epochs achieved an accuracy of 32.5%. The reason is that the context in these sentences is not particularly helpful and the semantic relation is only implicit in the text.

#### 4.3 DISCUSSION AND OWN CONTRIBUTIONS

In this chapter, we have proposed Onto.KOM as a minimally supervised framework for ontology learning. We showed that using word embeddings in combination with hierarchical clustering, proved to be quite effective for identifying the different ontological categories in a domain of knowledge. Moreover, the presented work showed that the concept of utilizing the concatenation (resp. the subtraction) of the embeddings of word-pairs as a basis for relation classification using CNN networks can provide better performance compared to the baseline without any manual features engineering.

Our experiments proved that using a combined classifier yields higher performance than using a separate classifier for each relation. In addition, a significant improvement in the performance can be noticed by increasing the number of training samples.

Finally, by evaluating our approach on AI-related trial data of SemEval 2016 Task 13, we showed the main merit of our approach: With the lack of a sufficient number of manually annotated sentences, Onto.KOM significantly outperforms state-of-the-art techniques. The superior performance of our approach is because we use the embedding concatenation between word-pairs. By training the structure using word-

pairs with similar context, it can predict the semantic relation of other similar word-pairs without their explicit co-occurrence in sentences.



## SEMANTICALLY ENHANCED MULTI-LABEL TEXT CLASSIFICATION

---

In this chapter, we address the second research challenge  $RG_2$  (see Section 1.2): *The high dimensionality of feature space and training overhead of existing approaches for multi-label text classification and the label imbalance in classification tasks*. Through our investigated related work we identified several research gaps in the existing methods for multi-label text classification. In the following sections, we introduce our proposed methods to address the identified research gaps. In Section 5.1, we present three multi-label datasets used in this work. Then, we discuss our approach for feature selection using typed dependencies in Section 5.2. The goal of our approach is to analyse how leveraging the semantic and syntactic dependencies between words can improve the quality of selected features. In Section 5.3, we study the feasibility of using deep learning structures for multi-label text classification. We carry out a comparative analysis of deep learning against traditional techniques for feature selection and text classification. In addition, we propose a new model with the goal to minimize the tedious supervised feature selection and classifier selection task in the context of a small dataset of long documents. In order to address the label imbalance challenge, we present our training-less classifier that leverages knowledge bases and word embeddings to transform the problem into graph matching problem in Section 5.4. Finally, we summarize our contributions and discuss future work in Section 5.5.

Parts of the work presented in this chapter have been previously published by the author of this thesis in [3, 5].

### 5.1 MULTI-LABEL DATASETS

In this section, the datasets used for evaluating the proposed methods are presented and analysed.

#### 5.1.1 EUR-Lex dataset

EUR-Lex<sup>1</sup> is a database providing free access to legal documents of the *European Union* including EU law, international agreements, and EU case-law in the 24 official EU languages. The dataset used for this work was collected by Menca and Fürnkranz [109] and is publicly available.<sup>2</sup> The provided dataset contains 19,348 documents in their English version. The English version of the documents contains also words in other languages. The number of non-English words varies between documents from a few words up to whole non-English paragraphs.

<sup>1</sup> <https://eur-lex.europa.eu/homepage.html?locale=de>

<sup>2</sup> <http://www.ke.tu-darmstadt.de/resources/eurlex>

The documents are indexed using three categorization schemes/labelsets as shown in Table 9. First, *EUROVOC*, a multidisciplinary thesaurus containing the activities of the EU. Second, *Directory Codes* are the official classification hierarchy of the EU of the Directory of Community in force. Third, *Subject Matters* which are descriptors based on the subject list of terms related to the documents. As seen in Figure 14 the labels

Labelset Name	# Classes	avg. Labelset size	Density	Distinct
Subject Matters	201	2.213	1.101	2540
Directory Codes	410	1.292	0.315	1615
EUROVOC	3956	5.31	0.134	16467

Table 9: Comparison between the different labelsets. Label density denotes the average number of labels per instance relative to the number of classes. Distinct displays the distinct label sets in the dataset. [109]

within the dataset are highly unbalanced with a few labels dominating the label space.

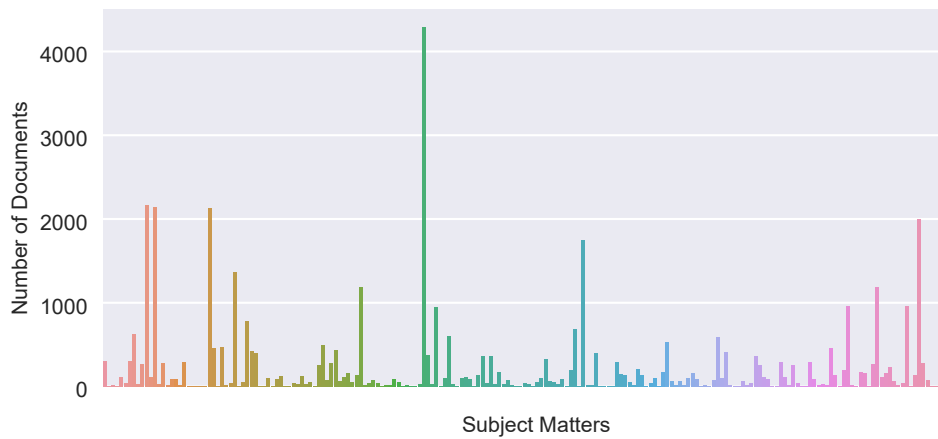


Figure 14: *Subject Matters* distribution in EUR-Lex dataset.

Furthermore, Table 10 shows that the length of the documents varies extremely with a mean of 2445 and median of 784. Besides, based on our analysis, the length of the documents ranges from 25 up to 329,165 words.

Input	Max	90 <sup>th</sup> -percentile	75 <sup>th</sup> -percentile	Median	Mean
words	329,165	5463	2034	784	2445
NPs	41,216	521	199	81	240
paragraphs	6,583	190	82	36	90

Table 10: Comparison of input lengths for different forms of input of EUR-Lex.

### 5.1.2 Reuters Volume Corpus I (RCV1)

RCV1 is an archive of over 800,000 manually categorized newswire stories made available for research purposed by Reuters, Ltd. [94]. The articles are written in English and stored in XML-format. Three different labelsets exist for RCV1: First, *Topic Codes* which describe the main subjects covered in the article. These labels are divided into four hierarchical groups: Corporate/Industrial, Economics, Government/Social and Markets. Overall there are 103 different topics used throughout the dataset. The topic code labelset is the one used during the evaluation based on RCV1 in this work. Figure 15 shows the distribution of the *Topic Codes* labelset. Second, *Industry Codes* which are assigned based on the type of business the article is about. The industry codes labelset is the largest RCV1 labelset and is grouped into ten sub-hierarchies. Third, *Region Codes* which cover both geographic locations and political/economic grouping. For this labelset no hierarchy exists.

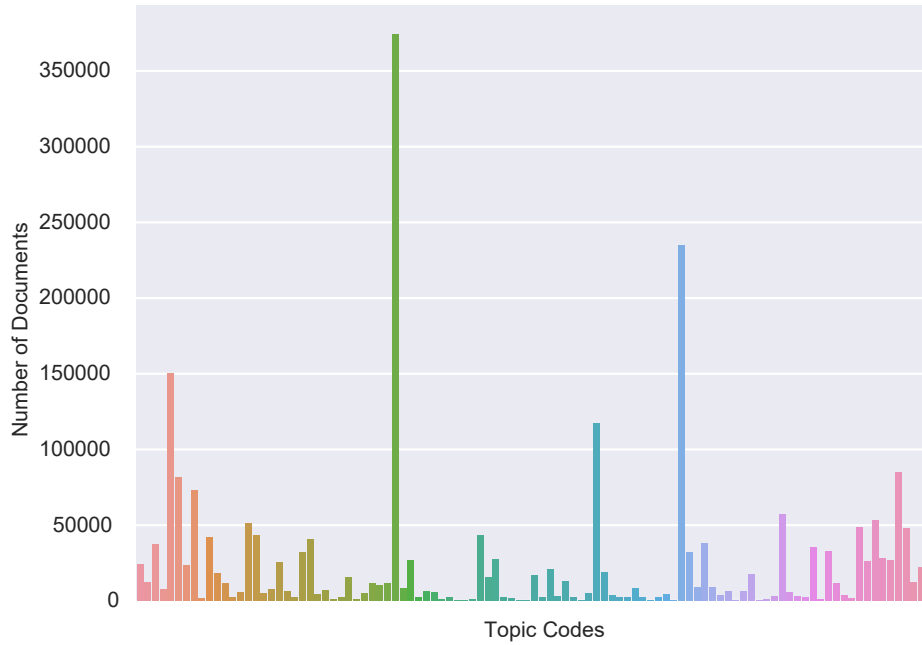


Figure 15: *Topic Codes* distribution in RCV1.

The RCV1 dataset exists in two versions RCV1-v1 and RCV1-v2. In RCV1-v2 documents that do not contain any topic code are removed, also topic codes not appearing in documents are removed. Lewis et al. restored ancestor topic codes for documents having a topic code but missing its superordinate topic code, adding additional 25,402 topics. For our experiments, we removed documents without topic codes as described before. Table 11 provides statistics on the number of words per document in the RCV1 dataset.

Input	Max	90 <sup>th</sup> -percentile	75 <sup>th</sup> -percentile	Median	Mean
words	6720	298	187	102	138

Table 11: Comparison of the document length in RCV1.

### 5.1.3 Reuters-21578 (ModApte)

The Reuters-21578 is a collection of Reuters newswire articles that appeared in 1987. Its original form contained 21,578 documents. Since then, numerous different splits have been suggested. In this work we use the modified Apte split (ModApte) provided by Natural Language Toolkit<sup>3</sup> (NLTK). This split contains 10,788 documents with 1,3 million words in total and a labelset size of 90. The split assigned 7769 to the training and 3019 documents to the test set. Figure 16 illustrates the distribution of the labels in Reuters-21578 (ModApte) dataset, while Table 12 provides statistics on the number of words per document.

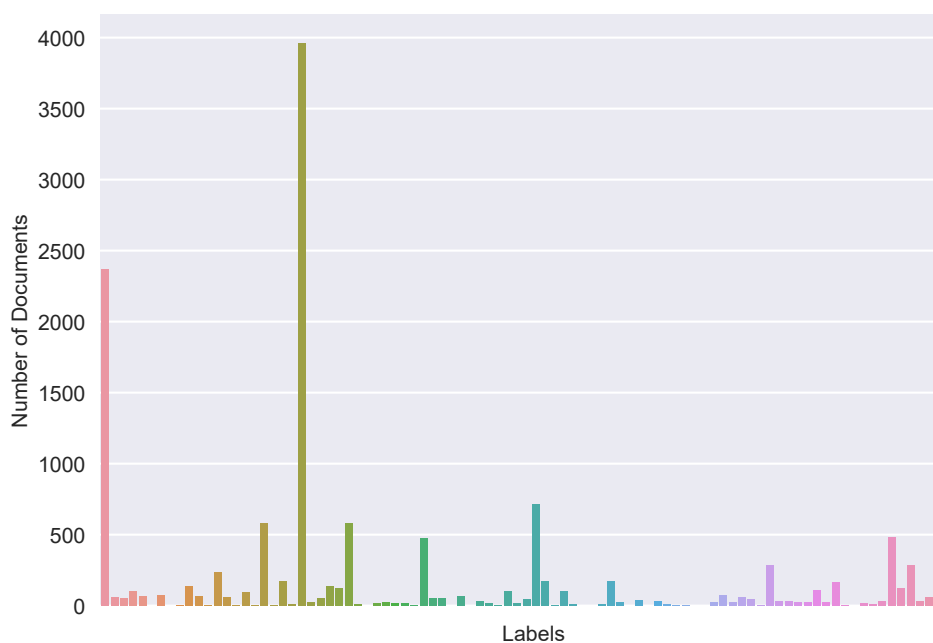


Figure 16: Label distribution in Reuters-21578.

Input	Max	90 <sup>th</sup> -percentile	75 <sup>th</sup> -percentile	Median	Mean
words	2650	316	178	97	144

Table 12: Comparison of the document length in Reuters-21578.

<sup>3</sup> <https://www.nltk.org/>

## 5.2 SEMANTIC-BASED FEATURE SELECTION USING TYPED DEPENDENCIES

As already mentioned in Section 1.2, a major challenge in text classification is the high dimensionality of the feature space. This feature space consists of all unique terms (words and NPs) occurring in a corpus. This can be hundreds of thousands of features on a small dataset, which is high for many algorithms. A wide range of statistical techniques have been proposed for weighting and selecting features in order to reduce the high dimensionality of feature space [40, 147]. Those techniques limited by losing semantic regularities of words as features and ignoring the dependencies and ordering between adjacent words. In addition, researchers have incorporated text semantics in feature selection by selecting NPs or n-grams as features, others tried to leverage external lexical databases mainly WordNet to enhance the performance by selecting relevant concepts. However, these methods are limited by the coverage of the used lexical databases. In this section, we propose a novel approach for incorporating the text semantics in feature selection by using typed dependencies between words to select and weight the features. Furthermore, we carry out a comparative study of our approach against a set of statistical and semantic-based techniques for feature selection.

### 5.2.1 Proposed Methodology

Reviewing related work for feature selection indicates that considering words order and context during the selection process can improve the classification performance (Section 3.2.1). In the proposed method, we incorporate context information and dependencies between words to select the features by using their typed dependencies. The process starts with a selection of relevant terms using a linguistic filter. Then we identify semantic and syntactic relations between term pairs based on the typed dependencies. By constructing an undirected *Term Graph*, different combinations of typed relations between the candidate terms can be used to select the features. The edges in this graph are labelled by the extracted relations between the terms. Figure 17 illustrates the proposed approach.

#### *Linguistic Filter*

First, we build a domain terminology by extracting all NPs in order to form the basis for our feature selection methods. The role of the linguistic filter is to recognize essential terms and filter out sequences of words that are unlikely to be concepts. Terms express concepts or restrictions of concepts, i.e., *algorithm* is a term used in *computer science*. In the linguistic component, the documents need to be preprocessed by a PoS tagger for marking up the words in a text, based on their context, as corresponding to a particular PoS, e.g., noun, preposition, verb, etc. Multi-word NPs like *Supervised Machine Learning* will be considered as one feature and are concatenated as *supervised\_machine\_learning*. Then, words that are unlikely to be part of concepts are excluded using a stop-words



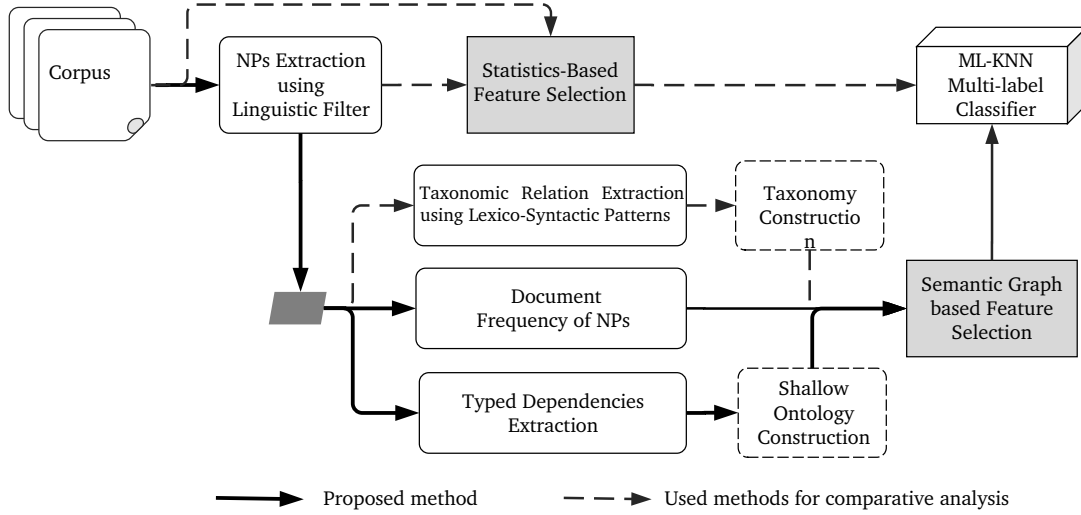


Figure 17: Block diagram of the proposed semantic-based feature selection method.

list. A combination of 3 linguistic filters is used to extract multi-word NPs that can reflect essential concepts.

- Noun Noun+
- Adj Noun+
- (Adj| Noun) + Noun

#### *Semantic and Syntactic Relation Extraction using Typed Dependencies*

A triple based representation such as, *abbreviated relation name (governor, dependent)* is mentioned as the typed dependency relation between words from the same sentence [35]. The term *abbreviated relation name* represents the type of dependency relation between any two words in the sentence, *governor* and *dependent* simply represents the position of the words within the sentence. For example, given the sentence “Bill is honest and polite”, one triple would be *conj(honest-3, polite-5)*, where *conj* is the relation between two elements connected by a coordinating conjunction. The parsing technique converts a sentence depending on the PoS tagging of words and the hierarchy of typed dependencies into a graph structure, then using this new representation, the syntactic relations on the sentence level can be identified to create the *Term Graph*. Figure 18 illustrates a sample sentence and its corresponding typed dependencies graph.

There are different types of dependency systems available such as, *Basic*, *Collapsed* and *Non-collapsed* [35]. We have considered the general collapsed dependencies, which represents the prepositions, conjunctions and other relative clause in a collapsed way to provide a direct relation between words. Using these dependencies, the syntactic features are defined and represented as triples of (*governor, dependent, relations*). A highlight in typed dependencies is *meronymy*. Meronymic relations are *part-whole* relations, where one entity is part or substance of another. Some dependencies like

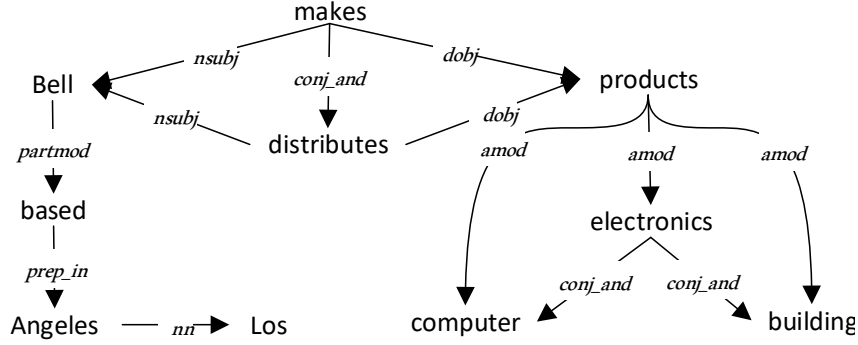


Figure 18: Typed dependencies for the sentence: *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.*[35].

*including, within, involving, inside, containing* imply these kind of relations. Berland and Charniak [8] used the dependency *of* to extract meronymic relations, however, we will not consider this dependency since it might lead to many weak relations.

#### Feature Selection using Typed Dependencies

We propose two feature selection techniques based on the associations between the extracted terms using the linguistic filter and the generated *Term Graph* using typed dependencies.

- **Concept Length:** Inspired by n-gram model, a NP with multi-words may have different meaning or contain more specific information than when it is treated separately. For example, the length of NP “French Financial Institutions” is 3. The NP “French Financial Institution” refers to a specific institution that is more specific and brings more information than “Institution”.
- **Typed Dependencies:** Typed dependencies between NPs represent the prepositional, conjunctive and verbal relations between words. We will try different combinations of these relations to identify the most representative set of features.

The *Term Graph* provides the candidate features and their relations. Later, the features can be selected based on two approaches, namely the DF and Concept Degree (CD). We define CD of  $NP_x$  as the number of NPs connected to  $NP_x$  from the *Term Graph* using typed dependencies, while DF represents the number of documents in which  $NP_x$  occurs. It is the simplest technique for feature selection. The basic heuristics behind using DF is that rare or infrequent words are non-informative for classification, however this is not correct in general. Respectively, two weighting techniques, namely binary weights, and TF-IDF, will be used to weigh the features with respect to the individual documents. TF-IDF is a global weighting method that reflects the importance of a word to a document in a corpus while Binary weighting is a local weighting method that reflects the presence of a word in a document.

### 5.2.2 Dataset and Experimental Settings

In the context of our comparative analysis, the EUR-lex dataset has been used (see Section 5.1.1). For the evaluation we used only *Subject Matters* categorization scheme. Stanford CoreNLP toolkit [103] was used in this work to perform the different NLP tasks (PoS, linguistic filter, taxonomic relations extraction and typed dependencies extraction). The used linguistic filter to extract single and multi-word terms resulted in 940685 distinct features (terms).

The carried out experiments aimed to compare the effectiveness of using *typed dependencies* for feature selection, taking into consideration the feature extraction based on taxonomic relations from our previous work [3] as a baseline. ML-KNN is used as the base multi-label text classifier. In order to be aligned with the baseline, the number of nearest neighbours was fixed during the experiments to  $K = 10$  and the number of features was fixed to 5000 features with 5-folds for cross-validation. For the features weighting, two techniques have been used, namely TF-IDF and binary weighting.

### 5.2.3 Evaluation Results

Five different evaluation steps were applied to analyse the different parameters of our approach.

#### *Evaluation of Statistical Methods*

In the first scenario, we compare the performance of our classifier by using all words as features against using only NPs. Three different approaches for feature selection were evaluated, namely selecting an equal number of features per label, proportionally to the label frequency and based on the average score for each feature over all labels. Selecting the feature based on the label frequency resulted in the best performance. For that it will be considered for further experiments. Figure 19 illustrates the different performance metrics for four statistical feature selection methods applied on the raw text documents after stemming and removing the stop words. Figure 20 illustrates the different performance metrics for using NPs as candidate features. The Concept-Document Frequency (C-DF) is the semantic-based feature selection method proposed in the baseline.

By comparing the performance of the different feature selection techniques, we can see that using NPs as candidates for feature selection instead of all words provides slightly better performance. That proves the importance of embedding simple semantic through the multi-word NPs to improve the performance. In addition, using NPs drastically reduces the computation costs by reducing the size of the feature space.

#### *Evaluation of Noun Phrase Length*

Table. 13 shows the effect of the length of NPs on selecting candidate features. We fixed the number of features to 5000 with TF-IDF weights. The comparison shows that choosing NPs with the length of 2 as features gives the best performance.

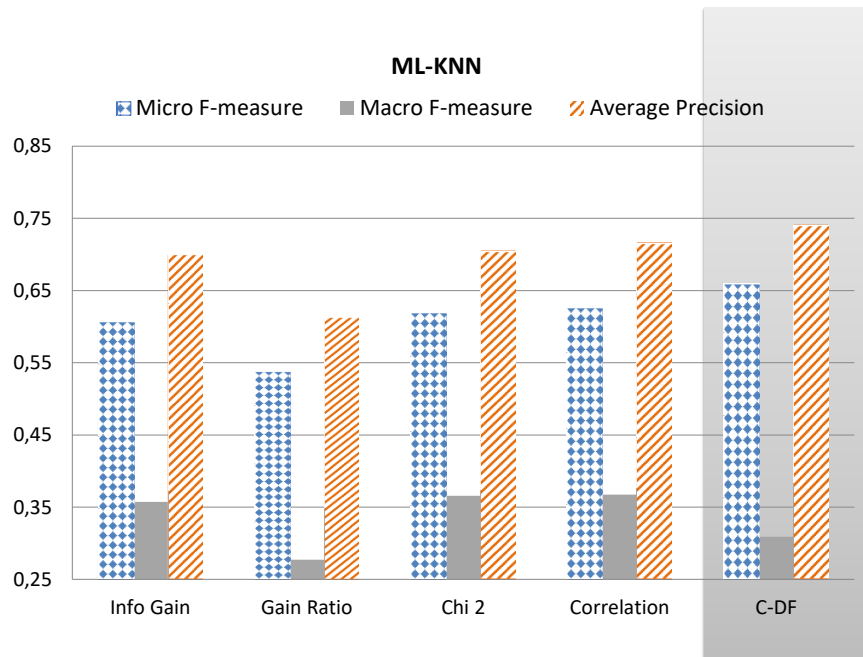


Figure 19: ML-KNN performance with the different statistical feature selection techniques using all words as features.

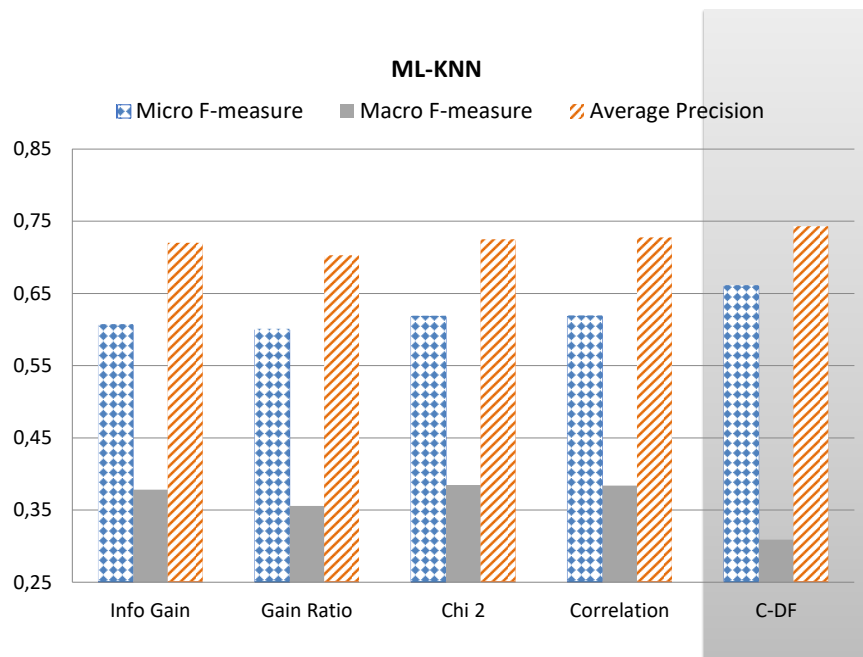


Figure 20: ML-KNN performance with the different statistical feature selection techniques using NPs as features.

Table 13: Evaluation results for feature selection by considering the length of NP.

NP Length	Hamming Loss	F1 <sub>micro</sub>	F1 <sub>macro</sub>	Average Precision
1	0.008	0.550	0.337	0.657
2	<u>0.007</u>	<u>0.579</u>	<u>0.352</u>	<u>0.661</u>
3	0.008	0.547	0.323	0.625
4	0.009	0.356	0.233	0.503
5	0.010	0.168	0.162	0.327

#### Comparison between DF and CD

In the third experiment, we compare two techniques to select the features from the *Term Graph* based on DF and CD as weighting techniques. Using the most common relation of with binary weights, we evaluated the performance impact of these two methods based on the number of features. Table 14 shows that DF results in better performance however the difference is small so both techniques can be used. Based on that, in the following scenarios we have fixed the selection technique to DF.

Table 14: Comparison between DF and CD for feature selection.

# Features	Document Frequency (DF)		Concept Degree (CD)	
	F1 <sub>micro</sub>	F1 <sub>macro</sub>	F1 <sub>micro</sub>	F1 <sub>macro</sub>
250	<u>0.627</u>	<u>0.355</u>	0.613	0.333
500	<u>0.642</u>	<u>0.3654</u>	0.635	0.362
1000	0.657	0.382	<u>0.661</u>	<u>0.388</u>
2000	<u>0.663</u>	0.394	0.659	<u>0.396</u>
2500	<b>0.665</b>	<u>0.401</u>	0.662	0.397
5000	<u>0.664</u>	<u>0.404</u>	0.660	0.401

#### Comparison between TF-IDF and Binary Weighting Techniques

In addition, we have investigated the quality of the extracted features based on two different weighting techniques, namely TF-IDF and Binary weights. The typed dependencies can form the *Term Graph* using propositions, conjunctions, and verbs as relation modifiers between the terms. In this step, we considered three sets of typed dependencies, namely using verb relations, using only meronomic relations and using all relations. Table 15 shows that using binary weights outperforms using TF-IDF as a weighting technique. This result is reasonable since the features are selected globally and not based on the document in hand. Also part-of verbs, reflecting verbal relations, i.e., *including*, *containing*, and *part-of*, slightly outperforms the other combinations of propositions, conjunctions and verb typed dependencies. Moreover, it outperforms the baseline too.

Table 15: Comparison between feature weighting techniques.

Types Dependencies	Binary Weights		TF-IDF Weights	
	$F1_{\text{micro}}$	$F1_{\text{macro}}$	$F1_{\text{micro}}$	$F1_{\text{macro}}$
verb relations	0.654	0.399	0.596	0.376
meronymy relations	<u>0.665</u>	<u>0.405</u>	0.594	0.371
all relations	0.659	0.392	0.585	0.369

*Comparison against the Baseline*

Table 16 shows the comparison between the baseline method [3], which considers the taxonomic relations, and our approach using typed dependencies for different number of features. The results indicate that using only the typed dependencies has slightly better performance compared to the baseline with regard to  $F1_{\text{micro}}$ , however, it outperforms the baseline with regard to the  $F1_{\text{macro}}$ . Considering the number of features, reducing the number to 2500 or 3000 can achieve nearly as good results as with 5000.

Table 16: Evaluation results against the baseline for the meronomic typed dependencies over different numbers of features.

# features	Hamming Loss	$F1_{\text{micro}}$	$F1_{\text{macro}}$	Average Precision
250	0.007	0.628	0.354	0.705
500	0.006	0.641	0.366	0.719
1000	0.006	0.655	0.382	0.736
2000	0.006	0.665	0.396	0.744
2500	<u>0.006</u>	<u>0.667</u>	0.400	<u>0.746</u>
3000	<u>0.006</u>	0.665	0.399	0.745
5000	<u>0.006</u>	0.663	<u>0.404</u>	0.744
<i>baseline</i> [3]	0.006	0.664	0.316	0.742

Overall, we showed that using typed dependencies can improve the quality of selected features in multi-label text classification. The proposed method is corpus-oriented and provides a global measure for feature selection disregard of the labelset. The main merits of this approach can be summarized by the better performance compared to other techniques, and the significant reduction in the computation costs.

### 5.3 DEEP LEARNING FOR MULTI-LABEL TEXT CLASSIFICATION

The results from the aforementioned approach align with the fact that up until now there are no standardized procedures to select the feature selection and transformation techniques and to determine the number of features. This is also applied to the selection of classifiers. Deep learning may play a key role in addressing these challenges. Deep learning models for NLP tasks leverage word and document embeddings in order to project the input into a reduced feature space that preserves the semantic similarity between features. In addition, deep learning-based classifiers are capable to capture local and global semantic features, to model the label's correlation and to establish a mapping between the meaning of a label and a document.

The majority of the deep learning methods, used for multi-label text classification, assume relatively large datasets of short text blocks or fixed input length. However, for many real-world applications, a relatively large number of labelled training samples is not available. Typically traditional classifiers tend to perform better than deep learning models when only a small amount of training data is available [175]. Also, the document length might vary from a few words to thousands of words in real world applications. This observation was the starting point for our investigations. We compare different deep learning methods and hyperparameters. We use the EUR-Lex dataset as it does not fulfil the commonly assumed properties. Our results thus provide important information for a suitable method selection and for hyperparameter settings.

#### 5.3.1 Comparative Analysis Settings

Our experiments study the effectiveness of using text embeddings and deep learning for multi-label text classification by analysing three different aspects, namely feature selection and document transformation techniques (see Section 2.2.2), more precisely the text embedding methods since we focus on semantic-based transformation methods, and the structure of the used deep network model. Figure 21 presents the experimental settings in this work.

##### *Form of Input (Granularity)*

A document is composed of multiple paragraphs, a paragraph consists of sentences and a sentence is formed by a sequence of words. All these structures could be used as input for the text embedding. We focus on how these different granularities of the input and the selection of text embedding methods might affect the performance of different neural networks compared to classical approaches. Several input forms can be fed to the deep neural networks:

- **Words:** Words are represented by Word2Vec or FastText representations as input. In order to be able to feed documents into the neural network, they all have to be of the same size. Therefore we truncate the documents up to a specific maximum document size corresponding to the average document length. If

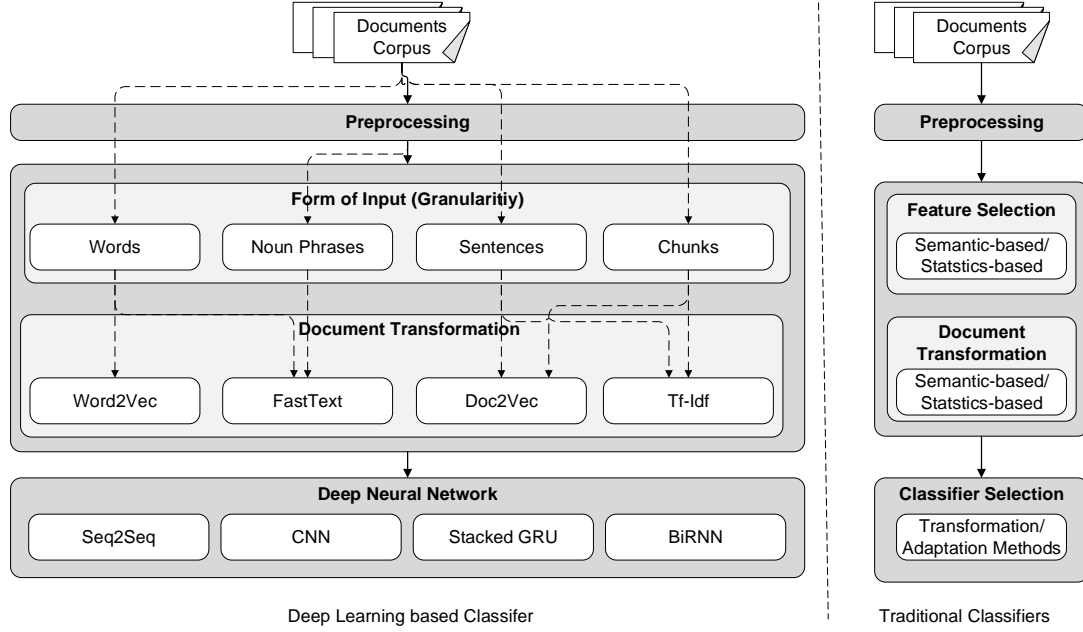


Figure 21: Workflow of the experimental scenario.

documents are shorter they are padded. FastText is used in this work because word vectors for multi-word terms that do not occur in the preprocessed corpora as a full match can be created using the subword information so that a meaningful representation of the terms can be generated.

- **NP:** To lower the maximum document length, but still preserve words containing rich information, NPs can be used as an input. In order to extract single and multi-word NPs from the document corpus, we use the linguistic filter presented in Section 5.2.1. Afterwards, FastText embeddings for the newly composed words are looked up, making use of FastText’s ability to process subword information.
- **Sentences:** In order to lower the document size further while maintaining its semantics, we consider sentences represented by Doc2Vec vectors as input. To ensure a minimum of information in a Doc2Vec vector we only consider sentences containing more than four words. It is most likely that shorter sentences do not carry enough information to infer Doc2Vec vectors which contain sufficient information to be well-performing inputs.
- **Chunks:** By assuming unstructured documents, we can not distinguish between paragraphs as a logical unit. For that, we split a document into  $n$  chunks of equal length. For each chunk, a Doc2Vec vector is inferred. The document is represented as the concatenation of chunk vectors.

#### Deep Learning Models

We perform hyperparameter optimization for four deep learning structures CNN, Seq2Seq, BiRNN and *Stacked GRU* (see Section 2.5.5).



On one hand, CNNs identify local features in large structures and combine those features to produce a fixed-size vector representation of the input. CNNs assume a fixed size input and predict the corresponding fixed-size outputs. On the other hand, RNNs are able to process sequential information. In our work, we consider the two most common cells, namely LSTM and GRU.

### 5.3.2 Analysis Results

In the experiments conducted, we used the EUR-Lex dataset with the subject-matters labelling scheme for the hyperparameters optimization and the comparative analysis with the-state-of-the-art techniques. In order to prove the transferability of our approach, we further analysed our best performing model using two other datasets, namely RCV1 [94] and Reuters2578 [6]. In addition, we have used the Stanford CoreNLP toolkit to perform the different text preprocessing tasks. 80% of the documents are used for training and 20% for test. The training set, used to optimize the deep neural networks, is further splitted into a 10% validation and a 90% training sets. All experiments are reported on the development set unless another set is mentioned.

#### *Doc2Vec Model*

In the case of using Doc2Vec as document representation method, the corpus used to generate the vectors and the length of the vectors also have an influence on the quality of the classification. Therefore, we first consider this aspect. We train the Doc2Vec model using three different corpora, namely EUR-Lex, Wikipedia dump <sup>4</sup>, and Deutsch Welles news <sup>5</sup> which includes around 76k articles. To analyze the influence of the vectors on the classification quality we use a ML-KNN classifier with a default number of nearest neighbours of  $k=10$  and we use the three different Doc2Vec models separately for input representation.

The Doc2Vec model generated using the Wikipedia dump showed the best performance (compare Table 17) as it is the largest corpus used. This means that training a Doc2Vec model on a relatively larger dataset would improve the quality of the generated document vectors and thus improve the basic classifiers too. Table 17 shows that increasing the vector size to 500 and then to 800, did not result in large improvements in the performance of the classifier. Therefore, we use Doc2Vec vectors generated using Wikipedia with 300 dimensions throughout the following experiments.

#### *Deep Learning Classifier Structure*

We perform several experiments in order to optimize the hyperparameters of the deep learning structure using Stacked GRU because it has reported the best performance in our experiments. The Stacked GRU has three GRU cells with the same number of hidden units. We examined in total six different hyperparameters: Dropout, batch size, number of hidden units, learning rate, number of epochs and number of chunks

<sup>4</sup> <https://dumps.wikimedia.org/enwiki/20190220/>. [Online; accessed 05-April-2019]

<sup>5</sup> <http://dw.com>. [Online; accessed 01-April-2019]

Table 17: Comparison between Doc2Vec models.

Corpus	Dimensions	$F1_{\text{micro}}$	$F1_{\text{macro}}$
EUR-Lex	300	0.300	0.082
Deutsche Welle	300	0.602	0.289
Wikipedia	300	0.674	0.326
Wikipedia	500	0.676	0.334
Wikipedia	800	0.675	0.334

used as input. To construct the vector representation of the chunks we chose Doc2Vec since it is capable of representing chunks as fixed-size vectors while preserving their nuanced semantics [90].

#### Stacked GRU Hidden Units:

In the first step, we determine the optimal number of hidden units used in the GRUs. For this purpose, we fix the other hyperparameters. Each model is trained for 100 epochs with a batch size of 64 and a learning rate of 0.0001. The drop out threshold fixed to 0.5 and the number of chunks per document  $n$  is fixed to 10. As Table 18 shows, after 512 hidden units the  $F1_{\text{micro}}$  score only increases marginal, but  $F1_{\text{macro}}$  profits significantly from a larger number of hidden units. Despite the fact that the 2048 hidden units used in the GRU, provide the best results and an even higher amount of hidden units might result in better results, we chose 1024 hidden units going forward as a trade-off between performance and computational complexity.

Table 18: The influence of the number of hidden units on Stacked GRU with chunks as input.

Number of Hidden Units	$F1_{\text{micro}}$	$F1_{\text{macro}}$
128	0.607	0.111
256	0.744	0.298
512	0.773	0.392
1024	0.778	0.429
2048	0.780	0.451

#### Learning Rate and Batch Size:

Similarly, each model is trained for 100 epochs with drop out threshold fixed to 0.5 and the number of chunks per document  $n$  was fixed to 10. Table 19 shows that the influence of batch size and learning rate on the  $F1_{\text{micro}}$  performance of the network is not significant. But these parameters can have a higher impact on the  $F1_{\text{macro}}$  performance. Based on these results we used a batch size of 128 and a learning rate of 0.0001 as best parameters for the following experiments.

Table 19: The influence of batch size and learning rate on the Stacked GRU’s performance with chunks as input.

Batch size	Learning Rate	$F1_{\text{micro}}$	$F1_{\text{macro}}$
64	0.0001	0.778	0.429
64	0.001	0.763	0.419
128	0.0001	0.782	0.445
128	0.001	0.775	0.419
128	0.005	0.762	0.379
256	0.0001	0.786	0.418
256	0.001	0.777	0.449
256	0.005	0.767	0.405

#### Fully Connected Layer:

The Stacked GRU model features a fully connected layer (FC) in between the Stacked GRU and the output layer. In this section, we evaluate the influence of different numbers of hidden units (HU) in the GRU and hidden units in the fully connected layer (FC HU) on the performance.

Table 20 shows that the model with the highest number of hidden units in the GRU layer and fully connected layer performs the best. It is to note that the GRUs with a low number of hidden units are almost able to close the gap in the performance with the help of a fully connected layer. This fully connected layer is a lot cheaper in terms of computational cost than adding additional hidden units to the GRU. In further experiments, we use RNN HU = 1024 and FC HU = 4096.

#### Number of Chunks:

We also evaluate the influence of different chunk sizes on performance. Figure 22 shows that  $F1_{\text{micro}}$  score declines after about 15 chunks. The highest  $F1_{\text{macro}}$  score is at 10 chunks. When the number of chunks  $n$  increases above 10 the performance decreases. We assume this is due to the fact that the chunks start to carry an insufficient amount of information as the chunk size becomes smaller. Therefore we use 10 chunks in the following experiments.

#### Number of Epochs:

Figure 23 shows that the  $F1_{\text{macro}}$  reaches its maximum after 100 epochs. This happens despite the fact that the loss is still decreasing after 100 epochs as shown in Figure 24. Therefore we use 100 epochs in the following experiments.

Table 20: The influence of the fully connected layer size on a Stacked GRU with different numbers of hidden units (RNN HU).

RNN HU	FC HU	F1 <sub>micro</sub>	F1 <sub>macro</sub>
256	128	0.723	0.259
256	256	0.761	0.333
256	512	0.778	0.388
256	1024	0.792	0.430
256	2048	0.798	0.465
256	4096	0.799	0.484
512	128	0.770	0.351
512	256	0.787	0.396
512	512	0.800	0.454
512	1024	0.803	0.454
512	2048	0.803	0.478
512	4096	0.800	0.498
1024	128	0.780	0.380
1024	256	0.794	0.433
1024	512	0.801	0.470
1024	1024	0.805	0.476
1024	2048	0.802	0.489
1024	4096	<u>0.799</u>	<u>0.495</u>

*Effect of Document Length*

We also examined the influence of the document length on training the Stacked GRU. We split the EUR-Lex dataset into two datasets of almost equal size based on their document length. Every document shorter than 449 words, corresponding to the median of document length, is considered as a short document. This resulted in 9683 short and 9655 long documents. These two datasets were then split into a training and test set using a 80/20 ratio. We used the optimized hyperparameters of the Stacked GRU.

Table 21: The influence of document length on the Stacked GRU's performance with chunks as input.

	F1 <sub>micro</sub>	F1 <sub>macro</sub>
Long Documents	<u>0.797</u>	<u>0.473</u>
Short Documents	0.746	0.366

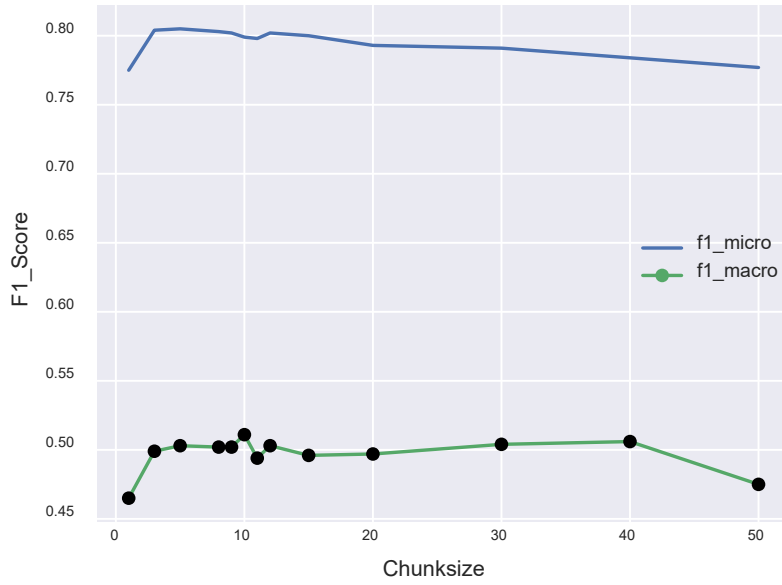


Figure 22: Performance of Stacked GRU for different numbers of chunks.

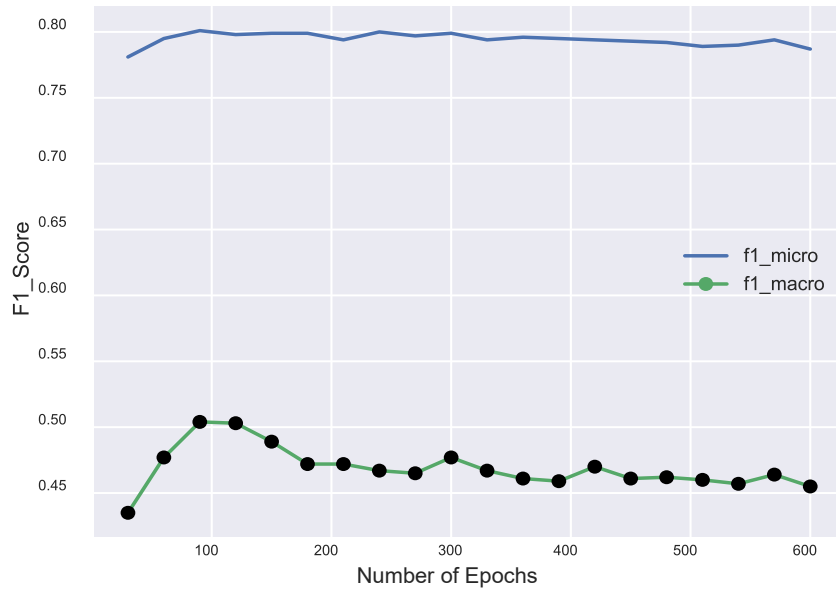


Figure 23: Performance of Stacked GRU based on the number of training epochs.

Table 21 shows that the Stacked GRU performs better for long documents. This is related to the number of chunks. By fixing the number of the chunks, the smaller the document the less information can be represented per chunk.

#### *Feature Selection, Document Transformation and Deep Learning Models*

After a detailed analysis of the suitable deep learning structures, we have further examined various combinations of feature selection and document transformation

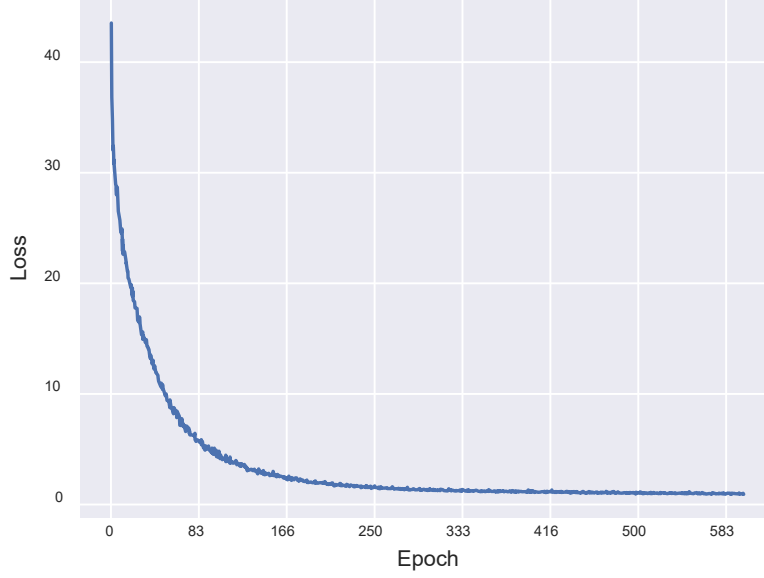


Figure 24: Training loss of Stacked GRU based on the number of training epochs.

methods and deep learning models. The results are shown in the first and second parts of Table 22.

To prove that semantic-based document transformation techniques like Doc2Vec provide a reasonable alternative for traditional feature selection techniques, we compare the results of an ML-KNN classification approach using multiple feature selection techniques (results 1-7 in Table 22) against the features provided by Doc2Vec (result 8). Overall, Doc2Vec features provide acceptable results, even if not the best, while requiring less computational complexity.

We started our experiments with different deep learning models, namely CNN, Seq2Seq and *Stacked GRU* (results 9-16 in Table 22). For BiRNN (result 17) we did our experiment only using chunks since they achieved the best results in all of the experiments. Using chunks as input reduces the complexity of the deep network and accordingly pushes its performance on smaller datasets. Overall the best approach is using *Stacked GRU* and *chunks* (result 16). To prove that by using semantic-based document transformation methods, deep learning structures can achieve state-of-the-art performance on a relatively small dataset, we compare the results with an approach using the best deep learning model *Stacked GRU* and TF-IDF (result 18). It can be seen, that the best approach of the four deep learning structures outperforms this approach.

#### *Comparison with Different Traditional Classifiers*

To prove that using deep learning structures can perform better than traditional classification approaches, we introduce further traditional classifiers (results 19-23) chosen for their distinct classification procedures. They are trained using their best-performing features TF-IDF. None of the approaches performs better than the best setup (result 16) of *Stacked GRU* using *chunks* based on Doc2Vec as input. Such a setup can provide bet-

Table 22: Comparative analysis of using deep learning models against the state-of-the-art techniques for the test set of EUR-Lex dataset.

#	Classifier	Input	Features	$F1_{\text{micro}}$	$F1_{\text{macro}}$
1	ML-KNN	Document	Tri-grams	0.758	0.398
2	ML-KNN	NPs	TF-IDF	0.759	0.409
3	ML-KNN	NPs	C-DF	0.664	0.316
4	ML-KNN	NPs	Info Gain	0.605	0.378
5	ML-KNN	NPs	Gain Ratio	0.599	0.355
6	ML-KNN	NPs	Chi 2	0.617	0.384
7	ML-KNN	NPs	Correlation	0.617	0.383
8	ML-KNN	Document	Doc2Vec	0.674	0.326
9	CNN	Words	FastText	0.223	0.169
10	CNN	Chunks	Doc2Vec	0.736	0.372
11	Seq2Seq	Words	FastText	0.716	0.308
12	Seq2Seq	NPs	FastText	0.659	0.248
13	Seq2Seq	Sentences	Doc2Vec	0.528	0.158
14	Seq2Seq	Chunks	Doc2Vec	0.721	0.318
15	Stacked GRU	Words	FastText	0.474	0.228
16	Stacked GRU	Chunks	Doc2Vec	<u>0.795</u>	<u>0.495</u>
17	BiRNN(LSTM)	Chunks	Doc2Vec	0.735	0.347
18	Stacked GRU	Chunks	TF-IDF	0.736	0.443
19	ML-KNN	Document	TF-IDF <sub>top5000</sub>	0.584	0.389
20	BPMLL	Document	TF-IDF <sub>top5000</sub>	0.027	0.140
21	BRkNN	Document	TF-IDF <sub>top5000</sub>	0.654	0.352
22	HOMER	Document	TF-IDF <sub>top5000</sub>	0.664	0.475
23	Clustering-B	Document	TF-IDF <sub>top5000</sub>	0.618	0.475

ter performance while eliminating the process of selecting a suitable dimensionality reduction technique and the corresponding classifier.

### 5.3.3 Approach Transferability

We further analyzed our approach using two other datasets, namely the RCV1 [94] and Reuters2578 [6] datasets. The experiments were conducted using the same configurations for the Stacked GRU and using chunks as input. The Doc2Vec model was trained on RCV1 for both experiments since Reuters-21578 is not large enough.

We compared the Stacked GRU using the RCV1 dataset against the best-reported performance on this dataset [23]. Similarly to the baseline, 80% of the documents are used for training and 20% for testing. This leads to 643,538 training and 160,884 test samples. The baseline used the top 1000 TF-IDF words according to their DF as features and BR as a classifier. Table 23 shows that our approach outperforms the baseline in terms of  $F1_{\text{macro}}$ , while it has a lower  $F1_{\text{micro}}$ .

Table 23: Comparison against the baseline using the test set of RCV1.

Classifier	Features	$F1_{\text{macro}}$	$F1_{\text{micro}}$
BR	TF-IDF <sub>top1000</sub>	0.687	<b>0.853</b>
Stacked GRU	Doc2Vec	<b>0.693</b>	0.825

In order to evaluate the structure on the Reuters-21578, we used the modified Apte split. This split contains 10,788 documents with a labelset size of 90. We used the predefined modApte split of 7769 training and 3019 test samples. Table 24 summarizes the performance against the baseline [23]. The proposed model has lower performance. This is because of the low number of training samples, but more detailed tuning of the hyperparameters would potentially improve the performance more.

Table 24: Comparison against the baseline using the test set of Reuters-21578.

Classifier	Features	$F1_{\text{macro}}$	$F1_{\text{micro}}$
CC	TF-IDF <sub>top1000f</sub>	<b>0.395</b>	<b>0.879</b>
Stacked GRU	Doc2Vec	0.274	0.722

#### 5.3.4 Summary

Overall, we showed that the distributed representations of documents like Doc2Vec provide a reasonable alternative for the traditional feature selection techniques. Besides, in the context of a small dataset of long documents, we proved that deep learning structures could perform better than traditional multi-label classification approaches while eliminating the process of selecting a suitable dimensionality reduction technique and the corresponding classifier. The significant difference in  $F1_{\text{macro}}$  and  $F1_{\text{micro}}$  over all the classifiers, including the proposed structure, indicates that the label imbalance problem persists, and more work is required in this direction.



#### 5.4 ONTOLOGY-BASED TRAINING-LESS MULTI-LABEL TEXT CLASSIFICATION

The results of the comparative analysis shown in Table 22 indicate that traditional classifiers tend to perform better for more frequent labels, our experiments proved similar characteristics for deep learning structures. In addition, multi-label datasets with millions of instances are increasingly common, which makes building and updating the classifiers an extremely tedious and time-consuming process. Based on the used algorithm, adding new labels might require re-training of the whole model.

In the following, we address these challenges and improve on previous research by proposing a novel training-less multi-label text classifier. We transfer the classification problem to a graph matching problem: the ontology effectively becomes the classifier. Thus, there is no need for a pre-trained classifier. The classification process is based on measuring the similarity between graph ontologies representing the labels and the main thematic entities representing the topics covered in a document. The label ontologies are mostly derived from a domain ontology. The method performs fairly across all the labels and proves less sensitivity to imbalanced datasets. Moreover, the solution presented in this work can be easily extended to incorporate new labels in the classification process with a minimized effort.

##### 5.4.1 Proposed Methodology

The proposed system is composed of four main components. The *Domain Ontology* which represents the knowledge contained in the problem domain. It is built based on the document corpus as well as a set of existing external lexical knowledge bases. A *Label Ontology* which represents the concepts related to a specific label. The ontology for each label is extracted using both the domain ontology and the documents corpus. The *Document Representation Module* which converts a text document to a set of topics. Finally, the *Matching Module* that converts the classification task into a matching between the topics representing a document and all the label ontologies.

##### *Domain Ontology Construction*

Figure 25 illustrates the workflow of developing the domain ontology. In order to build a comprehensive ontology we combine two existing lexical databases, namely *WordNet*, and *ConceptNet* to build a first ontology. *ConceptNet* [97] provides many non-taxonomic relations such as *has property*, *is used for* and *located near*. The semantic relations extracted from *Wordnet* are *part-meronym*, *substance-meronym*, *synonyms* and *hyponyms*. While, the extracted semantic relations from *ConceptNet* and *Yago* are *hyponyms*, *derived from*, *form of*, *has a*, *part of*, *manner of* and *synonyms*.

Since the lexical databases are static and have small coverage of concepts for particular domains, we expand the generated ontology by extracting additional relations using a lexico-syntactic pattern-based approach. We crawl new semantic relations from *Wikipedia* and 100,000 scientific papers published by *ACM*. First, a linguistic filter recognizes essential candidate concepts and filter out sequences of words that are un-

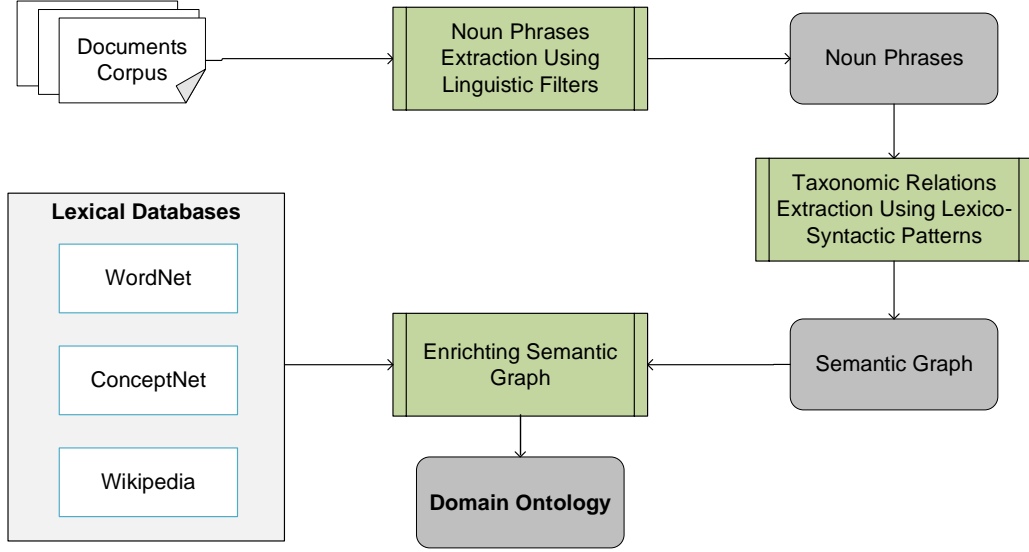


Figure 25: Workflow of developing the domain ontology in the proposed approach [5].

likely to be concepts. In order to extract single and multi-word NPs from the corpus, we use the linguistic filter presented in Section 5.2.1.

Then, we use the six Hearst lexico-syntactic patterns [64] to identify potential taxonomic relations. By this, we enrich the first ontology with around 113,000 new taxonomic relations. We keep the relations occurring at least 3 times to guarantee a high precision of the extracted relations. This threshold was defined based on a manual inspection of a random sample of the retrieved relations.

#### *Label Ontology Construction*

Figure 26 illustrates the workflow to develop label ontologies. We distinguish between two types of labels, namely *non-concrete labels* and *concrete labels*.

*Non-concrete labels* are the labels that are not found in the domain ontology. In order to find representative features for non-concrete labels, statistical-based feature selection methods are used in order to find the most representative NPs for each label. As shown in Figure 26, we extract NPs from the corpus. Then, a statistical filter is applied after that, in order to select the representative NPs/features. There are different approaches to analyse the dependencies between the NPs and the labels, namely IG, GR, chi-square statistic, and correlation, as presented in 3.2.1.

Labels that are found in the domain ontology are called *concrete labels*. The domain ontology is an adequate source for building the ontologies for the concrete labels. The label ontology is a subgraph/sub-ontology of the domain ontology. It is formed by selecting the concepts in the domain ontology with direct semantic relation to the label in hand.

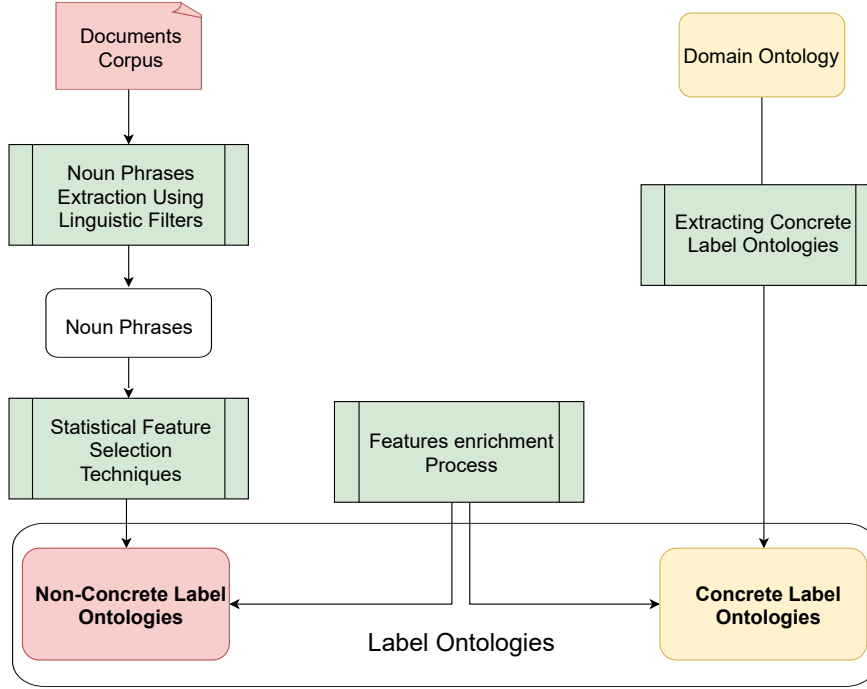


Figure 26: Workflow for label ontologies development [5].

#### *Label Ontology Enrichment*

Concrete label ontologies and non-concrete label ontologies form the target label ontologies. Both types can be extended by adding semantically similar concepts using *Word Embeddings* as illustrated in Figure 27.

Based on the characteristics of word embeddings that semantically-related words are close in the vector space [2], the generated word embeddings vectors are used to enrich the ontologies with semantically-similar words. Using the same corpus from Wikipedia and ACM papers used for the domain ontology construction, a FastText [11] model is trained and the generated word vectors are used. We add any concept, from the domain corpus, with high cosine similarity to a concept already existent in the ontology. We set a hard threshold of 0.9 for the cosine similarity which reflects words with similar context or meaning. This threshold is selected based on our previous work for synonym extraction (see Section 4.1).

#### *Document Representation Module*

In order to match a document against a label ontology, the main topics contained in the document should be extracted. Figure 28 illustrates the workflow of the document's main entities/topics extraction and the matching process.

A text document in the dataset might cover one or more topics. There are multiple options to approach topic modeling. Statistical approaches can be carried out for that purpose. On the one hand, the TF-IDF approach can identify the most important words in the document according to corpus-level measure. The Term Frequency (TF) and

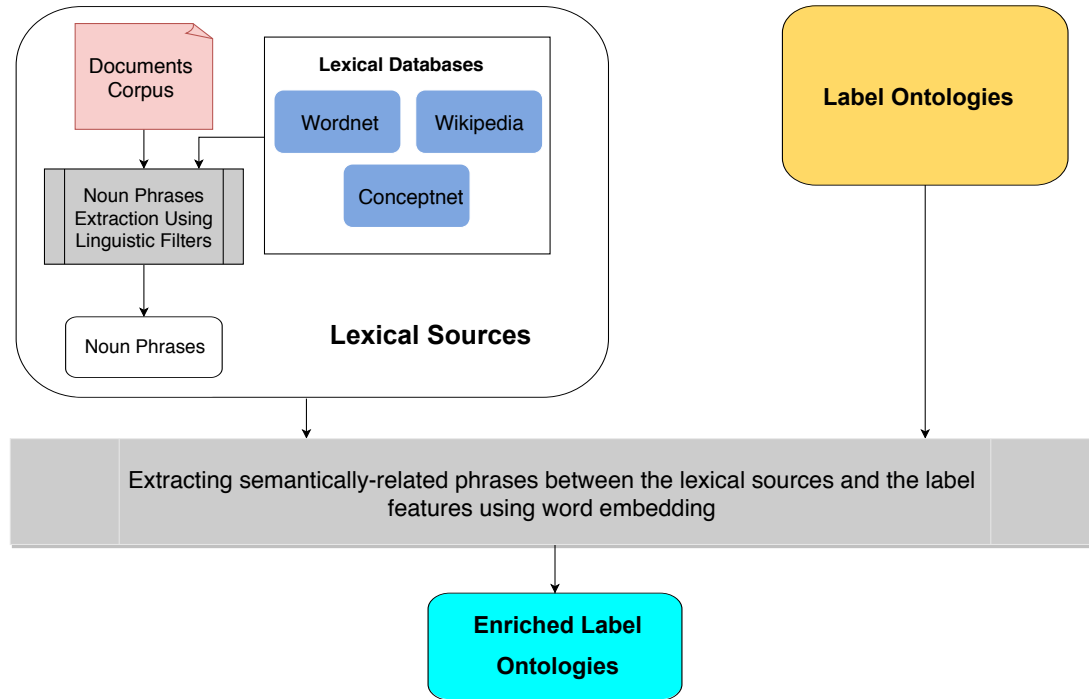


Figure 27: Block diagram for label Ontology enrichment process of label ontologies.

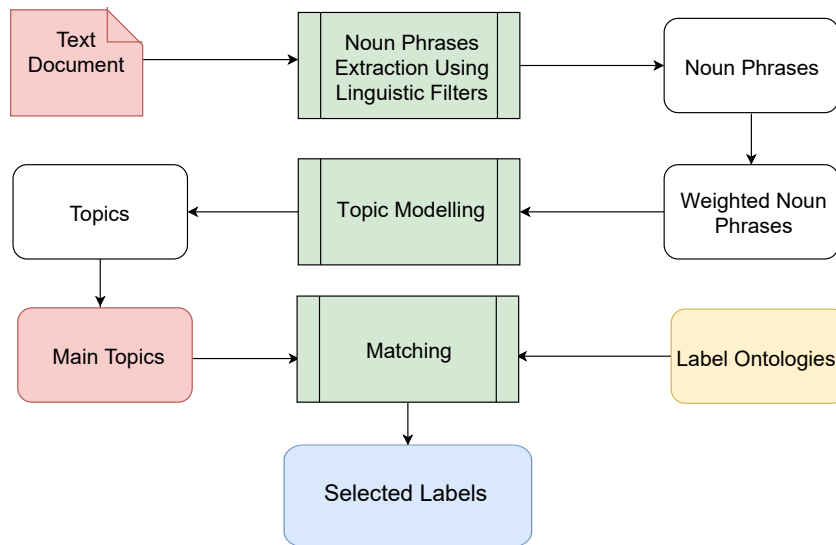


Figure 28: Workflow of document's main topics extraction and the matching process.

the Rapid Automatic Keyword Extraction (RAKE) approaches can identify the most important words in the document according to document-level measures. On the other hand, semantic approaches can be followed such as converting the text document to a graph of connected concepts using the domain ontology. A topic represents a set of words in a similar context, thus a method that correlates the importance of different words under the same semantic context should be used to define the weight of a topic.

In addition, the topic's weight or importance with regard to a specific document is proportional to the number of topics covered in the same document.

As shown in Figure 28, we first extract the NPs of a text document using linguistic filters. After that, a weighting technique is triggered to weigh each NP in order to measure the importance of the NPs with regard to the document. Both TF-IDF and TF were calculated. Then, the NPs are converted to word vectors using a trained FastText model. Having the representing vectors, a naive clustering procedure is applied to group the NPs into topics. Vectors which are similar to each other are clustered together into one set. At this point, each set is considered to represent one topic of the document. This was assumed because the vectors in the set are representing all the NPs (concepts) that are semantically similar.

After extracting the topics, a topic selection process is needed to filter out noisy and less informative topics. Topic selection methods were applied to rank the topics based on their importance to the document. Before ranking the topics according to their importance, certain rules are applied to remove noisy topics. For example topics that are composed of one NP with one occurrence in the document, topics that are not in English and topics including groups of numbers, etc. The first topic ranking step is based on TF-IDF. In the second step of the topic modelling procedure, we weighted the NPs using TF-IDF and TF. To rank the topics we perform the following procedure:

1. The average TF-IDF weight is calculated among the TF-IDF weights of the NPs of the document.
2. The topics which do not have at least one NP with a weight above the average is removed.
3. The total sum of TF-IDF is calculated for each topic according to the following formula:

$$\text{Imp}(\text{Topic}) = \sum_{i=1}^{i=n} \text{TF-IDF}(\text{NP}_i) \quad \text{where } \text{NP} \in \text{Topic} \quad (20)$$

4. The topics are ranked based on the TF-IDF calculated in the previous step.

The topics can also be ranked using the TF technique instead of TF-IDF following the same steps. Only One step needs to be changed. The weighting formula in Step number 3 is changed to be :

$$\text{Imp}(\text{Topic}) = \sum_{i=1}^{i=n} \text{Tf}(\text{NP}_i) \quad \text{where } \text{NP} \in \text{Topic} \quad (21)$$

### *Matching Module*

The Matching module is the module that performs the actual classification. For a document to be classified, the document representation module outputs the thematic topics of the document. Thematic topics are sets of NPs that are ranked to form the

most representing topics of the document. We have different label ontologies. A direct matching process is triggered to match the main topics (set of NPs) of the document to each label ontology (concrete or non-concrete). If there are NP/concepts which are included in the set of topics and a label ontology, then the label is considered to be a label for the document. How many concepts, matched between the topics and the label ontology, are necessary is a subject of the evaluation.

#### 5.4.2 Evaluation Results

We conducted experiments to empirically optimize the different configurable parameters of our approach. We compared the performance of our proposed approach against the best performance achieved using the deep learning approach presented before in table 22. The same split is used with 80% documents for training and 20% for test. The proposed solution has different parameters to be configured. They are analysed in the evaluation:

- *Correlation threshold for non-concrete labels*
- *Label ontology enrichment*
- *Topics selection approach*
- *Number of topics per document*
- *Matching process*

##### *Correlation Threshold for non-concrete Label*

Creating ontologies for non-concrete labels requires a statistical measure for the feature selection. We have considered four widely used techniques, namely correlation, information gain, chi-squared statistic, and information gain ratio. On the one hand, the number of label features based on chi-squared, information gain ratio, information gain is proportional to the label frequency which makes them less favorable for building the ontology of non-concrete labels. On the other hand, Figure 29 shows the number of features per label using different thresholds for the correlation score. Using terms with a correlation coefficient of 0.15 or more, we notice that the average number of features representing less-frequent labels against the most common ones are fairly close. For that, the statistical measure used in the following analysis is *Pearson's correlation coefficient*.

##### *Enrichment of Concrete Label Ontology*

The concrete label ontology can be built using the domain ontology and the dataset itself using *Pearson's correlation coefficients*. We compare five different combinations of sources to build the ontologies of concrete labels using the *Domain Ontology*, the correlated NPs and the correlated terms. The same settings are applied here.

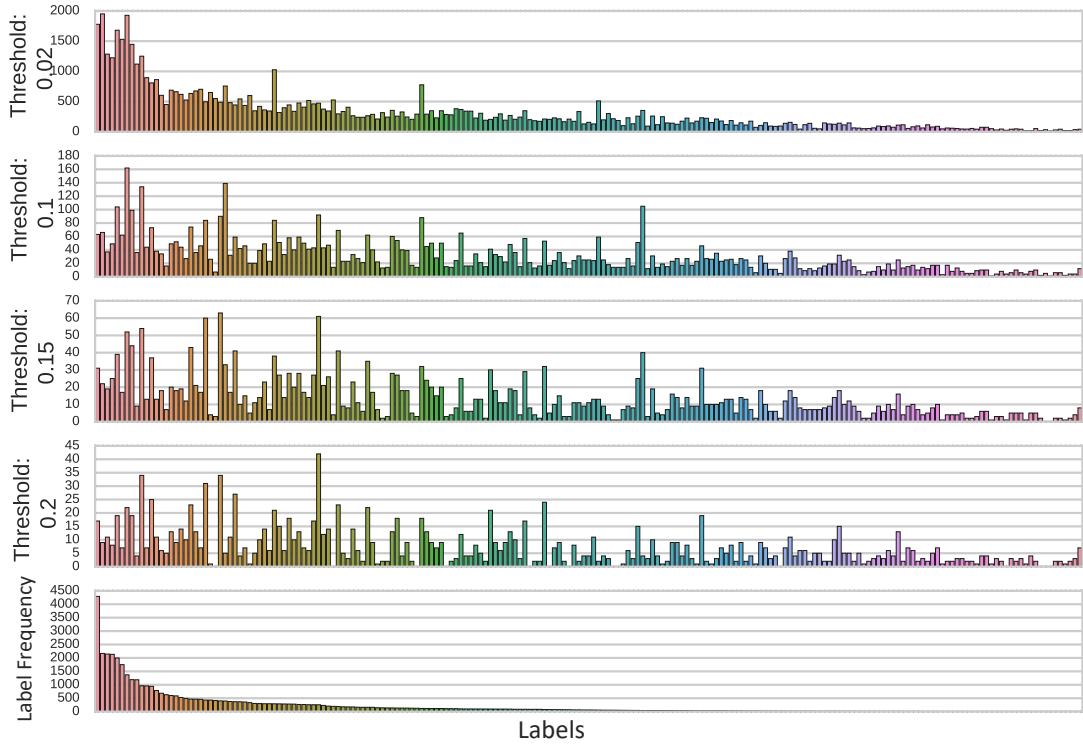


Figure 29: The number of features per label using different correlation thresholds for creating non-concrete ontologies.

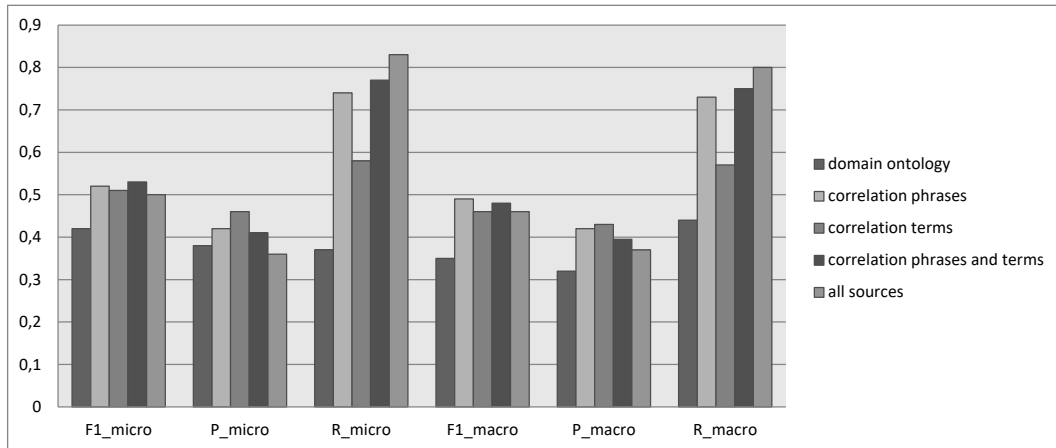


Figure 30: Using the different combinations of sources for concrete label ontologies.

Figure 30 illustrates the performance of the model over the different combinations. It shows that using a statistical measure to extract features from the dataset is better than depending on a domain ontology, which is contradictory to our assumptions. P\_Macro, R\_Macro, P\_Micro and R\_Micro in Figure 30 refer to  $Precision_{Macro}$ ,  $Recall_{Macro}$ ,  $Precision_{Micro}$  and  $Recall_{Micro}$  respectively.

For that, we analyse the concrete label ontologies taken from the domain ontology, and noticed that some concrete label ontologies have very few concepts or even no concepts in the domain ontology. Thus, in the next experiment, we consider 13 concrete labels whose ontologies have more than 45 related concepts. We analyse the classifier performance by using the domain ontology as a source for concrete label ontologies against using the correlated phrases as a source. Figure 31 proves that considering the domain ontology as a source for the label ontology is better than using correlation phrases. This is valid for labels that have enough concepts in their ontologies.

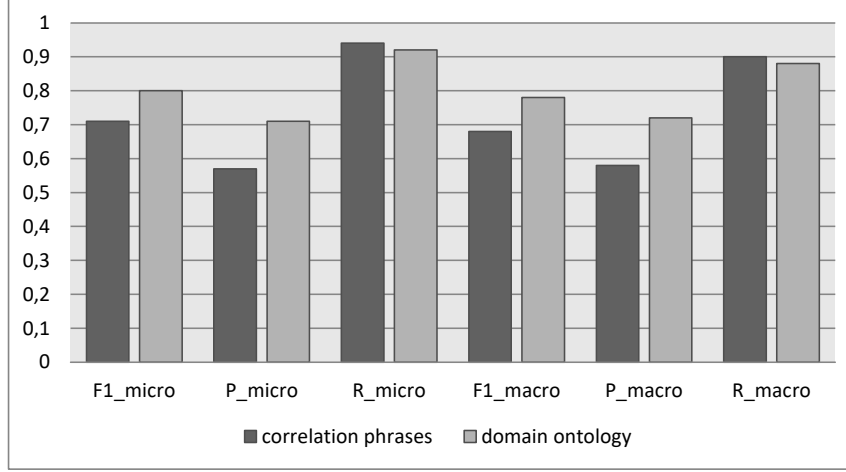


Figure 31: Using correlation phrases vs domain ontology for enriching concrete label ontologies.

In order to increase the coverage of the label ontologies, we conduct a feature enrichment process to extend the label ontologies. The basic idea is to add more features that are semantically related to the existing features. Table 32 illustrates the results using the same 13 concrete labels whose ontologies have more than 45 related concepts. It proves that extending the ontology with semantically related concepts improves the performance further.

#### *Topics Selection Approach*

Topics importance can be measured using TF-IDF or TF. Figure 33 illustrates the performance using both techniques as basis for ranking the topics representing a document. It is clear that using TF-IDF for measuring the topic importance is better than depending on TF.

#### *Number of Topics per Document*

After topic extraction, a subset of representative topics should be selected. Using TF-IDF for topic selection, we investigate the performance based on the number of selected topics. The threshold to be defined is the minimal number of topics required to represent a document. Figure 34 illustrates the model performance with regard to the



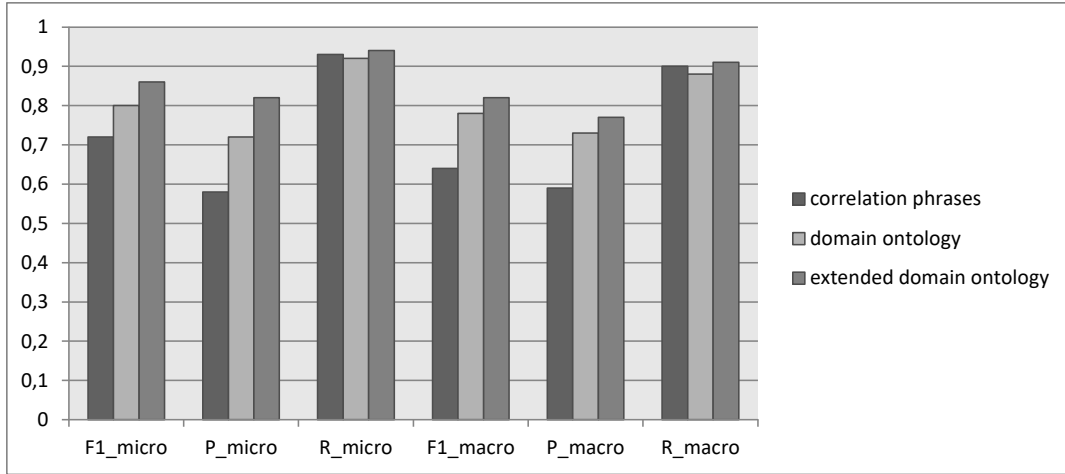


Figure 32: Using correlation phrases/domain ontology/extended domain ontology for features in concrete labels with few concepts in the domain ontology.

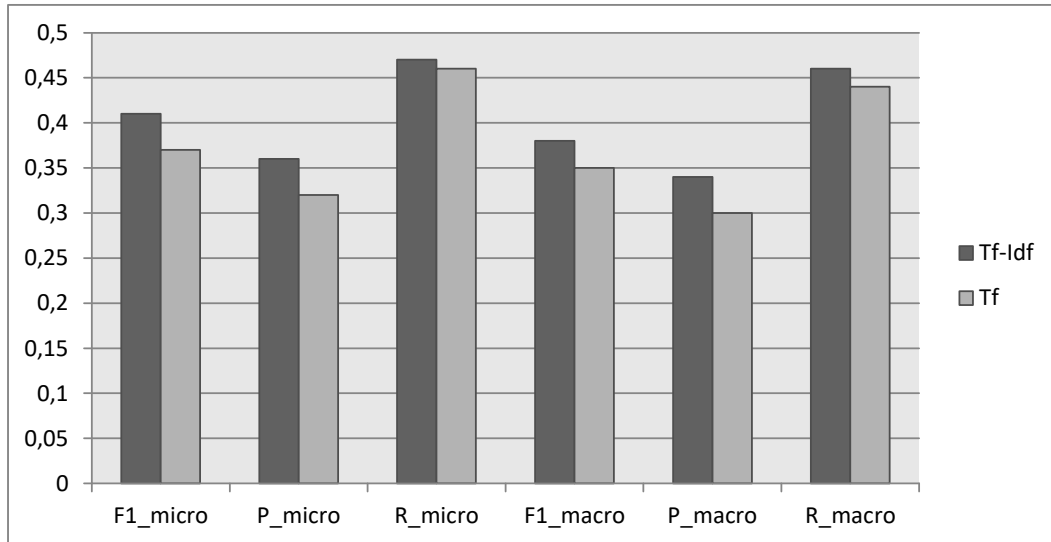


Figure 33: Using TF-IDF and TF as for ranking the topics representing a document.

number of selected topics. It is clear that the performance is decreasing by increasing the number of topics representing a document. The best performance reached by using the top 2 topics representing a document which conforms to the *label cardinality* of the dataset.

#### Matching Process Analysis

The matching process is done between a topic in a document and a label ontology. We analyse the required proportion of NPs in a topic set that should be covered by the label ontology to be considered as a match. Figure 35 illustrates the experimental

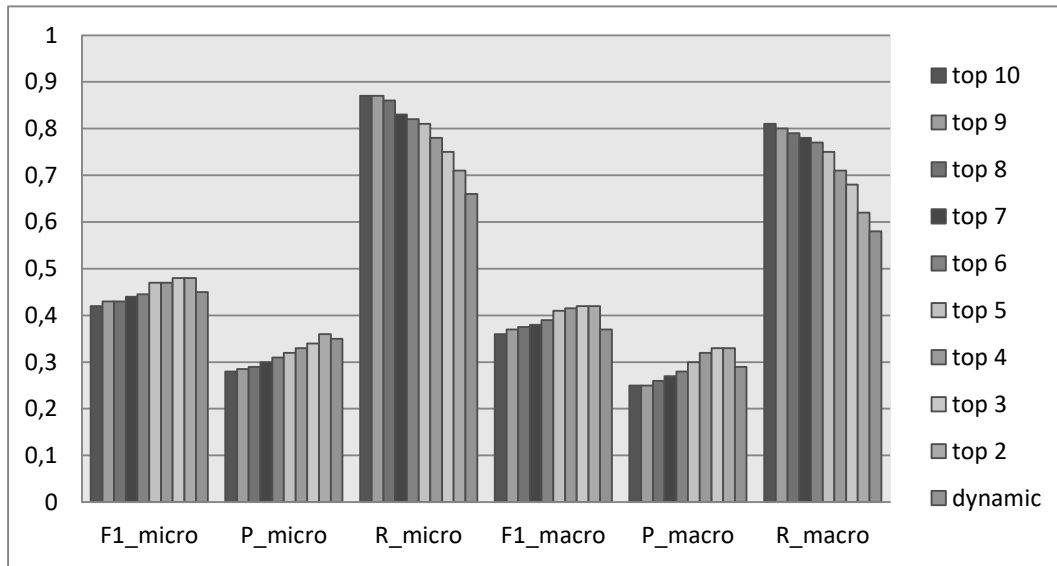


Figure 34: The performance based on the number of topics representing a document.

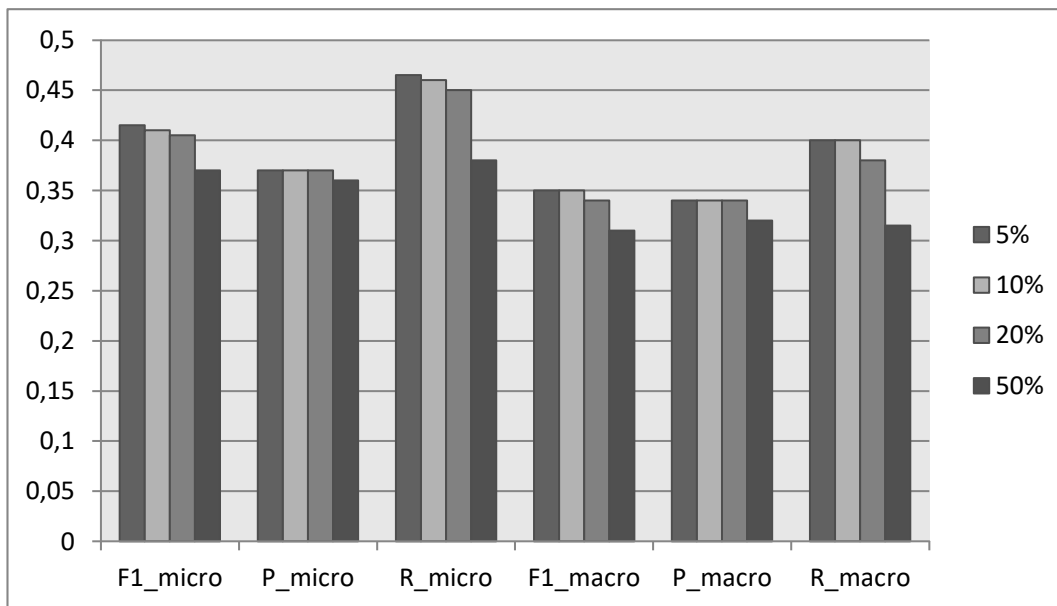


Figure 35: The performance based on the proportion of NPs in a topic set covered by the label ontology.

results. It indicates that using a matching threshold of 5% of the topic set is the best option for the matching decision.

### 5.4.3 Comparative Analysis

We compare this approach against the top performed classifiers from the previous section in Table 25 in terms of  $F1_{macro}$ . The proposed Stacked GRU with chunks outperforms all other approaches in terms of  $F1_{micro}$  and  $F1_{macro}$ . The methods used in the baseline (#2,3) perform better in terms of  $F1_{micro}$  and significantly lower in terms of  $F1_{macro}$  compared to the training-less classifier. This conforms with the fact that traditional classifiers tend to perform better for frequent labels. Our training-less classifier has a fair performance across all labels with close scores for both  $F1_{micro}$  and  $F1_{macro}$ . In addition to our proposed Stacked GRU with chunks and Doc2Vec, only two classifiers, out of 22 reported in this work, outperform our training-less classifier.

Table 25: Comparative analysis of training-less classifier against the baseline.

#	Classifier	Input	Features	$F1_{macro}$	$F1_{micro}$
2	ML-KNN	Document	Tri-grams	0.758	0.398
3	ML-KNN	Document	TF-IDF	0.759	0.409
12	HOMER	Document	TF-IDF <sub>top5000</sub>	0.664	0.475
13	Clustering-B	Document	TF-IDF <sub>top5000</sub>	0.618	0.475
21	Stacked GRU	Chunks	TF-IDF	0.736	0.443
<u>22</u>	<u>Stacked GRU</u>	<u>Chunks</u>	<u>Doc2Vec</u>	<u>0.795</u>	<u>0.495</u>
23	Training-less Classifier	Document	Ontologies	0.501	<b>0.468</b>

## 5.5 DISCUSSION AND OWN CONTRIBUTIONS

In this chapter, we proposed new methods to address the main challenges in multi-label text classification, namely, the high dimensionality of the feature space, the label imbalance and the training overhead. First, we presented a new method for feature selection by leveraging the context, order and dependencies between words. The model proposed uses only the typed dependencies between words on the sentence level to select the features. Using these dependencies, we can identify syntactic and semantic relations even with informal text blocks. Our experiments proved that using the typed dependencies between words can provide better performance compared to statistical and semantic-based approaches. In addition, our model significantly reduces the computation costs by relying on the shallow ontology for selecting and updating the features.

Second, we explored how deep learning structures can be used to minimize the tedious supervised feature selection task and classifier selection task in multi-label text classification in the context of a small dataset of long documents, where classical techniques tend to perform better. We proved that by reforming the input, deep learning structures can achieve state-of-the-art performance on a relatively small dataset.

In fact, our approach using chunks of text as input performs better than raw words, sentences or documents. Moreover, we showed that semantic-based document transformation methods like Doc2Vec provide a reasonable alternative for the traditional feature selection techniques introduced in Sect. 3.2.1. In addition, we demonstrated that using deep learning structures can perform better than traditional multi-label classification approaches while eliminating the process of selecting a suitable dimensionality reduction technique and the corresponding classifier.

Finally, the proposed methods, as mentioned earlier, do not overcome the label imbalance problem. For that, we have developed an ontology-based training-less classifier, which transfers the problem into a graph matching between a label ontology extracted using a domain ontology and the main topics representing a document. The designed classifier performed equally for frequent and less frequent labels, which proved that the built model overcomes the label imbalance problem of traditional techniques.



## COMBINING HETEROGENEOUS INFORMATION SOURCES FOR PERSONALIZED CITATION RECOMMENDATION

---

In this chapter, we address the third research challenge  $RG_3$  (see Section 1.2): *Providing personalized citation recommendation using multi-source heterogeneous information*. We tackle the problem of citation recommendation as a special case of document recommendation. In citation recommendation, we can distinguish between two types of information, namely content information, and citation information. Content information includes the content of publications, i.e., the full manuscript, the keywords, and the title. Citation information corresponds to other types of bibliography information including information about co-authorship, affiliation, the venue and year of a publication as well as information about previously cited papers by authors.

Through our presented related work, we identified several research gaps in the existing methods for citation recommendation. In the following, we propose two methods for personalized citation recommendation by leveraging citation information and content information. We investigate how combining these different heterogeneous information sources can improve the performance of personalized citation recommendation. Both approaches assume that the author's previous publications are given.

Section 6.1 presents the first approach, which incorporates two modules to build an ensemble. The first is a query-based recommendation module, named Q-DSSM, that suggests papers based on their semantic similarity to the query text. The proposed model learns hidden latent features using the DSSM structure [73] to implicitly measure the semantic similarity between a set of keywords representing the query text and the candidate documents under unsupervised conditions. The second is a graph-based ranking module that provides a ranked list of papers based on an algorithm that traverses a graph representing authors, papers, citation information, and content information of all candidate papers. The starting point for the traversing are the authors of the paper for which citations should be recommended, hereinafter referred to as query authors. The fusing of the recommendations provided by both modules forms the final recommendation list.

In our second approach (Section 6.2), we use the ARGA model [122] to integrate citation information (network structure) and content information into a unified framework. The ARGA model is used to represent the heterogeneous bibliographic information in a low dimensional, compact, and continuous feature space. Based on the proposed model, we can obtain graph embeddings of the heterogeneous bibliographic network. Using the graph embeddings in combination with the information about previously cited papers, we can provide personalized recommendations.

Parts of the following work have been previously submitted by the author of this thesis and accepted for publication in the book series *Advances in Analytics for Learning and Teaching*<sup>1</sup>.

## 6.1 PERSONALIZED CITATION RECOMMENDATION USING AN ENSEMBLE MODEL OF DSSM AND BIBLIOGRAPHIC INFORMATION

The proposed ensemble model consists of three main components, namely the *Query-based Recommendation Module*, the *Graph-based Ranking Module*, and the *Fusion Module*, as shown in Figure 36. The Query-based Recommendation Module uses a DSSM model to measure the semantic similarity between the query text, which is a document or a part of a document for which citations are searched, and the candidate papers. It provides a list of documents ranked based on their cosine similarity to the query text. The Graph-based Ranking Module uses the citation information, content information, and information about the previous publications of the query authors to recommend a ranked list of papers based on their relevance to the query text. Finally, the Fusion Module combines both recommendation lists to provide the final citation recommendation using a simple averaging technique.

In addition, an ontology is used as input for the *Query-based Recommendation Module*, which is created automatically once using a chain of NLP techniques and external knowledge bases. The ontology is used to filter out irrelevant NPs from the query text. It is a preprocessing step of the Query-based Recommendation module.

Multiple word and document representation techniques are used. Character-level trigram-based word hashing is used for the representation of the keywords extracted from the query text and the topics extracted from candidate documents. These representations will form the input for the DSSM model in the Query-based Recommendation Module. To represent the documents in the Graph-based Ranking Module we use Doc2Vec model. Doc2Vec represents text blocks in a reduced feature space as a vector of numbers [11]. Using the Doc2Vec model, documents with similar context will be close in the vector space. The FastText embeddings are used to represent words. As mentioned before, FastText is a form of word embedding capable of learning character n-gram representations in addition to the word itself. Thus it can take subword information into account. The Doc2Vec and FastText models are trained using the whole document corpus and Wikipedia dump.

In the following sections of this chapter, we explain the ontology creation procedure and the modules in detail.

### 6.1.1 Ontology Construction

The ontology consists of concepts contained in existing lexical databases and their semantic relationships. We combine different existing lexical databases, namely *WordNet* [112], *YAGO* [100] and *ConceptNet* [146] in order to build a basic ontology. The semantic

<sup>1</sup> <https://www.springer.com/series/16338>

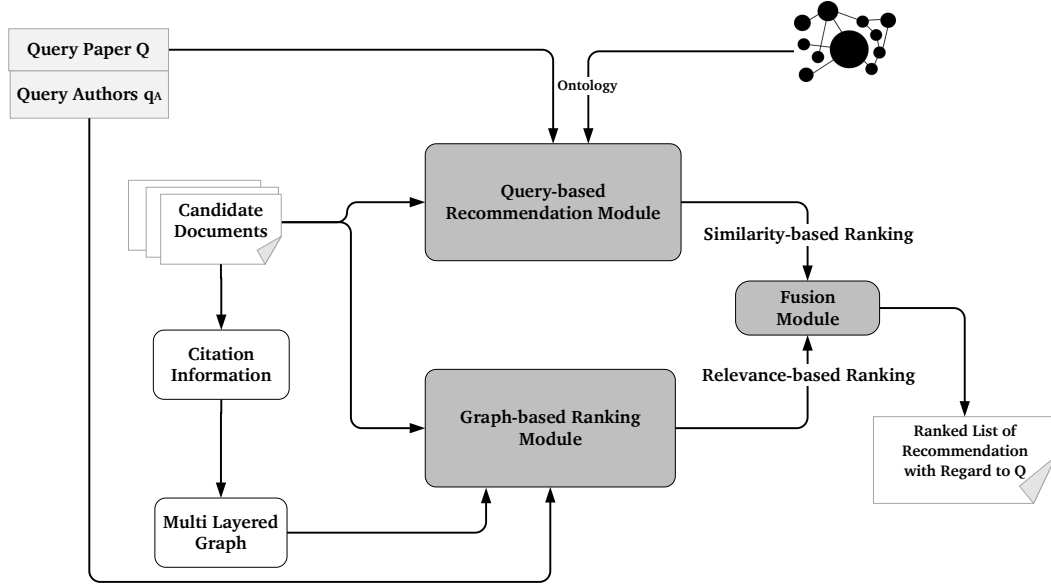


Figure 36: Block diagram of the personalized citation recommendation system.

relations extracted from *Wordnet* are namely, *Part-meronym*, *Substance-meronym*, *Synonyms* and *Hyponyms*. The semantic relations extracted from *ConceptNet* and *Yago* are namely, *Hyponyms*, *Derived-from*, *Form-of*, *Has-a*, *Part-of*, *Manner-of* and *Synonyms*.

Besides, we use the six Hearst patterns [64] to extract taxonomic relations from the corpus. By that, we enrich the basic ontology with around 113,000 new taxonomic relations. Ambiguous patterns might return correct as well as incorrect relations. Hence, we keep only the relations found at least three times in the corpus using our patterns to guarantee a high precision of the extracted relations. The ontology is extended further using the FastText model. We add any concept with high cosine similarity to a concept already in the ontology, as a potential synonym. We set a hard threshold of 0.9 for the cosine similarity, which reflects words with similar context or meaning by these we minimize the number of incorrectly selected concepts as synonyms. The threshold is selected based on an ontology enrichment approach using the synonym relation proposed in our previous work in [2].

### 6.1.2 Query-based Recommendation Module

This module ranks the candidate documents based on their semantic similarity to the query text (the whole manuscript, title or a paragraph) as input. It transfers the recommendation process to information retrieval space and takes only the content information into account. Contrary to traditional approaches for citation recommendation with poor usage of semantics, it adapts the state-of-the-art semantic similarity model DSSM to measure the similarity between the query text and the candidate documents. Figure 37 illustrates the process of generating recommendations, which consists of the following three sub-components:



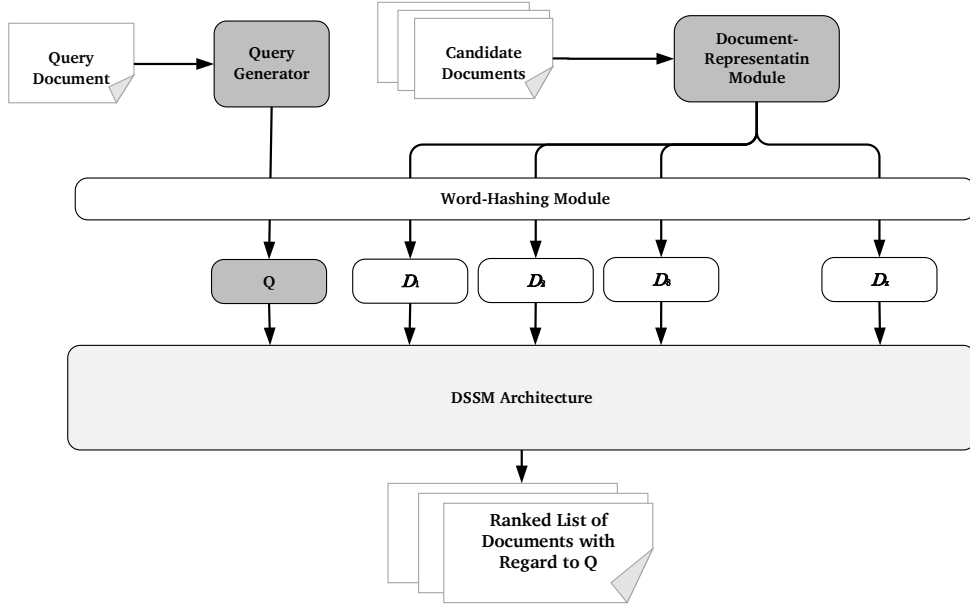


Figure 37: Block diagram of the query-based recommendation module (Q-DSSM).

The first component is the *Query Generator*, which takes the query text as input. This module extracts keywords which are most representative for the query document and are used as input to the DSSM. First, NPs in the query document which consist of two or more nouns are selected as candidate keywords using a combination of 3 linguistic filters (see Section 4.1.1). Single words are excluded as they might represent very common or high-level concepts that are less probable to represent a keyword. In order to select the keywords from the multi-word NPs obtained, we use RAKE algorithm: an unsupervised document-oriented keywords extractor [136]. RAKE weights the importance of each NP with respect to the whole document. Based on previous investigations, we use only the eight most highly weighted NPs. By matching the extracted keywords with concepts in our ontology, we remove all nouns that do not represent concepts such as names, universities, cities, countries, etc. The resulted list might still include concepts that do not represent the main topics of the document. To eliminate those concepts, we pass all extracted concepts through a word embeddings filter. By this, we remove all concepts that do not have a strong semantic relation with the other concepts extracted.

Second, the *Document Representation Module* represents each document as a set of topics. Firstly, the NPs are extracted using the same linguistic filters. Then using their *FastText* representations, we cluster the NPs using a density-based clustering algorithm called DBSCAN [47] with  $\text{min\_samples} = 2$  and  $\text{eps} = 0.5$  corresponding to a soft cosine similarity threshold between word-pairs. The vectors which are similar to each other, based on their cosine similarity, are grouped into one cluster. At this point, each cluster is considered to be representing one topic of the document. This was assumed because the vectors in a cluster are representing all NPs that are semantically similar.

Topics with only one NP are excluded. By this, we filter out most of the NPs that do not represent relevant topics, i.e., author names, typos, etc.

The third component is the DSSM model, which measures the semantic similarity between two text blocks. DSSM is a latent semantic model, with a deep neural network, that projects documents and queries into a common low-dimensional space. Therefore, the relevance of a document given a query is equivalent to their distance in that common space. Our proposed model is based on a modified version of the DSSM structure in [73] as shown in Figure 38. The structure is a fully connected neural network with two non-linear projection layers. The output of the DSSM is a vector with 120 features that represent the latent semantics of the input. The number of hidden units and layers is empirically selected.

The input for the DSSM model are the keywords extracted by the *Query Generator* using the query text and the representations of the available documents generated by the *Document Representation Module*. The model has a preprocessing input layer that converts the input queries and documents representations, using a *Word-Hashing Module*, into vectors of letter trigrams. Finally, a cosine-similarity layer is attached on top of the DSSM to measure the similarity between the query text and the candidate document representations. The output of this layer is a rank for each document regarding the semantic similarity to the query.

The DSSM should learn the semantic similarities between a query and its corresponding document. In order to train the DSSM in an unsupervised way, we generate synthetic query-document pairs from our training set by automatically extracting the keywords representing a document using the *Query Generator* to form a query and use the original document as query-document pairs. The negative query-document pairs are generated by selecting documents with low cosine similarity with the document corresponding to the query using their Doc2Vec representation.

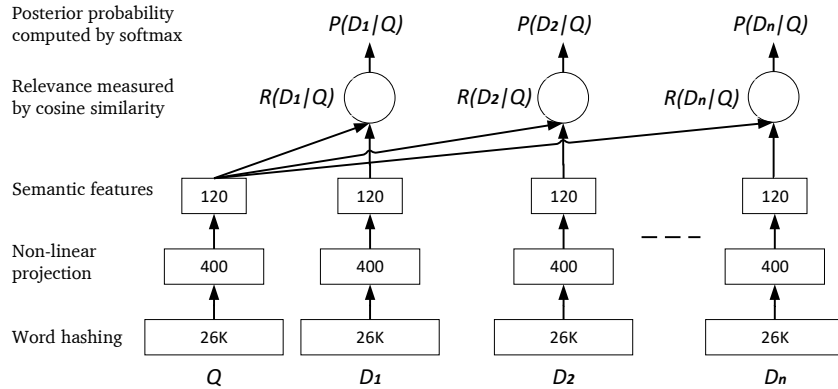


Figure 38: The proposed DSSM model structure.

### 6.1.3 Graph-Based Ranking Module

This module is based on two assumptions. The first assumption is that authors have a high interest in the preliminary work of their co-authors and in papers cited in the preliminary work of the co-authors. This is expressed in the algorithm described below by the fact that these papers are taken into account during the recommendation procedure. The second assumption is that authors work in relatively unchanging research areas, and therefore, publications in this area are potentially more relevant to them than publications from other areas. This is expressed in the algorithm described below by the fact that papers that are similar to those cited in the author's preliminary work are included in the calculation. These references are made by looking at the citations relations.

The first step in the module is to create a graph of all authors, papers, and citations in the papers contained in the paper corpus. As shown in Figure 39, it is a multi-layered graph. In the second step, this graph is traversed using the vertices of all current authors as the starting point following a relevance-based ranking algorithm as described below in Algorithm 1.

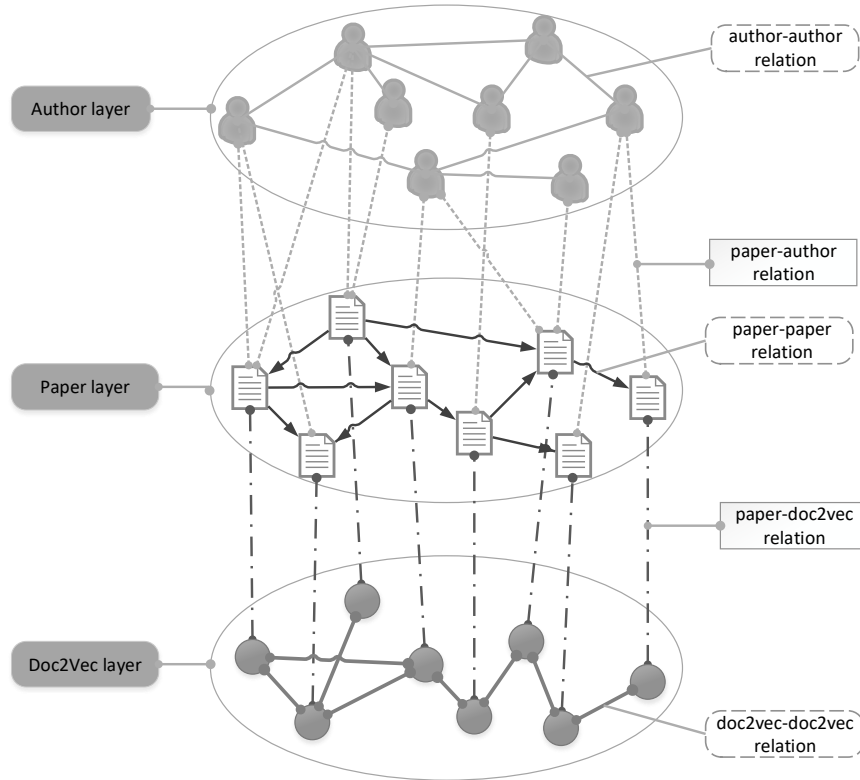


Figure 39: Multi-Layer graph modelling.

### Multi-layered Graph

In this work, we consider papers and authors as vertices of the graph. Papers and authors are modelled in own layers of the graph, based on citation information. The first layer contains all authors as vertices and the co-authorship relations as edges. The second layer contains all papers as vertices. An edge between two papers, A and B is drawn if paper A cites paper B. An intra-layer edge between an author and a paper indicates the authorship relation. In the third layer, each paper is represented by its *Doc2Vec* representation. The weighted edge between two of these representations is corresponding to the cosine similarity between the *Doc2Vec* representations. In this way, the content information is also taken into account. To limit the number of edges, only papers whose cosine similarity exceeds a given threshold are considered. We now formally present the definitions of the multi-layer graph as follows:

- **Author Layer:** The relations in this layer form a graph that describes the co-authorship between authors. For simplicity, the author graph is an undirected graph, which can be represented by a tuple,  $G_a = \{V_a, E_a\}$ , where  $V_a$  are the author vertices  $a$  and  $E_a$  the unweighted edges which represent co-authorship between two authors.
- **Paper Layer:** The paper layer is corresponding to the citation graph between the papers in the corpus. The citation graph is a directed graph, which can be represented by a tuple,  $G_p = \{V_p, E_p\}$ . Each vertex  $p$  in  $V_p$  represents a paper, and each edge in  $E_p$  is unweighted and represents citation between two papers
- **Doc2Vec Layer:** This layer considers the semantic similarity between the papers. Each vertex represents the *Doc2Vec* representation  $Vec_i$  of a paper  $p_i$ , and each edge between two paper representations represents the cosine similarity  $sim_{i,j}$  of the *Doc2Vec* representation of two papers  $Vec_i, Vec_j$ .

The paper-author relationship forms the connectivity between the Paper Layer and the Author Layer, while each paper in the Paper Layer is connected to its *Doc2Vec* representation in the *Doc2Vec* Layer.

### Relevance-based Ranking

In the relevance based ranking, the Multi-layered Graph is traversed to select and rank the papers, based on the similarity to the query text, which are recommended to an author. The vertices of all current authors in the graph are the starting point of the traversing. This creates a personalized recommendation for the authors. Algorithm 1 shows the overall ranking algorithm.

#### 6.1.4 Fusion Module

The *Fusion Module* fuses the rankings of both modules to result in the final recommendation list. For simplicity, we consider averaging the rankings of each paper from both

**Algorithm 1** : Graph-based Ranking

**Data** : ACL anthology network (ANN): The author set  $A$ , the paper set  $P$ , the content representation set  $Vec$ , the query text  $q_t$ , the query author  $q_a$ , similarity threshold  $sim_{Thr} = 0.3$

**Result** : Ranked list of papers as recommendation

```

1 begin
2   /* initialize an empty recommendation list S of papers      */
3    $S \leftarrow \{|P|, 0\}$ 
4   /* for all query authors  $q_a$                                 */
5   for  $a \in \{q_a \cap A\}$  do
6     /* for all previously published papers  $P_a$  of author  $a$     */
7     for  $p \in P_a$  do
8       /* calculate the similarity between paper  $p$  and query  $q$  */
9        $sim_{p,q} = \text{Similarity}(Vec_p, Vec_q)$ 
10      /* add the paper  $p$  to the recommendation list  $S$ , if not
11      contained and update the score of paper  $p$  in the
12      recommendation list  $S$  if the similarity is above the
13      threshold */
14      UpdateWeightS( $p, sim_{p,q} + (S(p))$ )
15      if  $sim_{p,q} \geq sim_{Thr}$  then
16        /* get all papers  $c$  cited in  $p$  */
17         $C \leftarrow \text{CitedBy}(p)$ 
18        for  $c \in C$  do
19          /* calculate the similarity between cited paper  $c$  and
20          query  $q$  */
21           $sim_{c,q} = \text{Similarity}(Vec_c, Vec_q)$ 
22          /* update the score of paper  $c$  in the recommendation
23          list  $S$  */
24          UpdateWeightS( $c, sim_{c,q} + S(c)$ )
25
26      /* add papers similar to these contained in their recommendation
27      list  $S$  */
28      /* for all papers  $p$  from the corpus which are not contained in
29      the recommendation list  $S$  */
30      for  $p \notin \{S \cap P\}$  do
31        /* for all papers  $p_1$  contained in the recommendation list */
32        for  $p_1 \in S$  do
33          /* calculate the similarity between paper  $p$  and paper  $p_1$  */
34
35           $sim_{p,p_1} = \text{Similarity}(Vec_p, Vec_{p_1})$ 
36          if  $w_{p,p_1} \geq 0.8$  then
37            /* add the paper  $p$  to the recommendation list  $S$ , if not
38            contained and update the score of paper  $p$  in the
39            recommendation list  $S$  */
40            UpdateWeightS( $p, sim_{p,p_1}$ )
41
42  return Ranked list of candidate papers in  $S$ 

```

modules. In this way, the model gives higher rank to papers that are previously cited by the query author and are more similar to the query text, while papers that never been cited before and which have low semantic similarity to the query text will be ranked lower.

#### 6.1.5 Dataset and Evaluation Results

The ACL anthology network (AAN) is a corpus of scholarly publications in Computational Linguistics [10]. The papers published in different venues from 1965 to 2013 were used as the experimental dataset. We removed papers with missing titles or abstracts, which resulted in 12,555 papers. Papers published between 1965 to 2012 were deemed as the training set (11,197 papers), and the 1,358 papers published in 2013 were used as the test set, similarly to Cai et al. [17] in their work.

The experiments carried out aim at studying the effectiveness of using the proposed model for personalized citation recommendation by analysing three different aspects, namely the input representation, the network structure of the DSSM, and the performance against the baseline. In addition, we analyse the influence of using the information about the past citations of the authors, which means we compare the personalized approach with a none-personalized approach.

#### 6.1.6 Q-DSSM Structure Optimization

Two major aspects were analysed concerning the DSSM structure, namely the number of hidden non-linear projection layers and the input encoders of the word-hashing trigrams. The DSSM structure was optimized by minimizing the cosine-similarity based loss over the training and validation sets. Two different encoders were used to represent the weighted word-hashing trigrams. The first one is Term Count (TC) as a document-oriented technique, and the second is TF-IDF as a corpus-oriented technique. As the input for the DSSM are character-trigrams, we calculate TC and TF-IDF on the character-trigrams. In all experiments, we use *ReLU* as the default activation function for neurons and applied batch normalization. *Adam Optimizer* was used as the optimizer. Additionally, we have used an initial learning rate of 0.0001 and trained our model with a batch size of 256 over 150 epochs.

Table 26 presents the cosine-similarity based loss of the Q-DSSM over four different structures and two types of encoders. Using TF-IDF to encode the weighted word-hashing representation of the query and documents reduces the loss over the training and testing sets compared to using TC. This can be justified by the fact that TF-IDF reflects the importance of the different trigrams for a specific document with regard to the whole corpus. Accordingly, using a shallower structure of two layers with 400 and 120 neurons respectively performed better compared to more complex structures, which is mainly related to the number of training samples that can support optimizing the DSSM as well as the number of trigrams representing the vocabulary. Consequently, in all further experiments, the first structure will be used with TF-IDF as the encoding method.

Table 26: Training and validation loss corresponding to the different configurations of the DSSM structure and the encoder.

DSSM Layers			Using TC		Using TF-IDF	
L <sub>1</sub>	L <sub>2</sub>	L <sub>3</sub>	Training Loss	Validation Loss	Training Loss	Validation Loss
400	120	-	1.0	24.5	0.1	0.6
1000	500	-	0.8	25.0	0.09	2.0
2000	1500	1000	1.5	38.1	0.15	3.1
2000	1500	500	1.3	23.0	0.08	2.7

#### 6.1.7 Personalized vs. Non-Personalized Recommendation

In this section, we analyse the influence of the personalized approach in comparison to content-based approaches. Hence, we compare our overall personalized approach (Personalized) with an approach which does not use the graph-based ranking but only the query-based recommendation (Q-DSSM). By this, the approach will be reduced to a non-personalized recommendation system that relies on the semantic similarity between the candidate papers and the query text using the Q-DSSM. For comparison with a simple approach (Doc2Vec) we calculate the cosine similarity between the Doc2Vec representation of the candidate documents and the query text to rank the candidate papers in the recommendation list.

Table 27 shows that the personalized recommendation is clearly superior to the non-personalized recommendation on the test set. Also, the query-based recommendation system shows better performance compared with a basic Doc2Vec approach, which proves that the quality of the vector representations provided by the DSSM is better.

Table 27: Personalized vs. non-Personalized recommendation systems.

Approach	Recall@20	Recall@60	Recall@100
Q-DSSM	0.256	0.334	0.415
Doc2Vec	0.196	0.319	0.382
Personalized	<u>0.268</u>	<u>0.419</u>	<u>0.494</u>

#### 6.1.8 Comparison with other Personalized Citation Recommendation Systems

Finally, we compare our approach with other state-of-the-art personalized recommendation systems using the ANN dataset and following the same split for training and testing as reported in the papers cited below.

- *The neural probabilistic model-based approach* [74] that jointly learns a distributed semantic representation of citation context and cited papers. The recommenda-

Table 28: Comparison with the baselines on AAN dataset.

Approach	MAP	MRR	Recall@20	Recall@40	Recall@100
HITS [120]	0.073	0.089	0.186	0.258	0.378
MultiLayer Graph Model [121]	0.091	0.108	0.209	0.275	0.396
Unified Graph Model [110]	0.108	0.117	0.221	0.286	0.410
LocDiSCern [19]	0.113	0.120	0.226	0.292	0.421
GloDiSCern [19]	0.112	0.119	0.225	0.290	0.418
Neural Probabilistic Model [74]	0.119	0.126	0.234	0.299	0.441
Mutually Reinforced Model [17]	0.126	0.137	0.242	0.331	0.479
Our Approach	<u>0.042</u>	<u>0.562</u>	<u>0.268</u>	<u>0.365</u>	<u>0.494</u>

tions for queries are provided by training a multi-layer neural network model to estimate the probability of citing a paper given a citation context.

- *DiSCern* [19] that uses only the citation information and preassigned keywords associated with the papers to retrieve relevant and diversified citations. Similarly to [17], we use the keywords in the keyword section of each paper in the AAN dataset; if no keyword section is found, a graph degeneracy-based approach is used to extract keywords. Both variations of this algorithm are considered, namely LocDiSCern and GloDiSCern.
- *HITS* [120] that which uses a set of extracted technical terms to generate a bipartite graph consisting of papers retrieved by those terms and the technical terms appearing in these papers. Then top-ranked papers are recommended to the author based on the HITS algorithm.
- *Multilayer Graph Model* [121] that builds a heterogeneous bibliographic graph using citation and content information. Then, a graph-based similarity learning method is used to provide a ranked list of recommendations.
- *Unified Graph Model* [110] that additionally incorporates co-authorship and venue information.
- *Mutually reinforced model* [17]: that uses a three-layered interactive clustering approach to cluster related vertices in the graph. Using the subgraph, generated by the clusters associated with each researcher’s needs, a list of recommendations is provided.

Table 28 shows that the proposed model significantly outperforms all other approaches in terms of MRR and MAP, which indicates a better ranking performance of the proposed model. On average, 56.2% of the top-ranked recommendations are correct. The table also shows a better performance concerning Recall@N.

The proposed model is characterized by the flexibility to integrate various content and citation information. However, it ignores the latent similarity between authors.



The starting point for traversing the graph in the graph-based ranking module are the authors of the query in hand; this means that publications from other authors, who have cited the same papers, will not be retrieved unless their publications are semantically similar. In addition to that, their publications will be ranked lower in the list of recommendations. In order to overcome this limitation, we propose in the next section another structure that encodes both citation and content information in a unified framework and represents them in a lower-dimensional space. Consequently, authors with similar citation and content information will be close in the feature space, which can address the aforementioned problem.

## 6.2 USING ADVERSARIALLY REGULARIZED GRAPH AUTOENCODER FOR PERSONALIZED CITATION RECOMMENDATION

Our second approach for citation recommendation incorporates two modules. The first is a generative adversarial bibliographic network module that learns an effective graph embedding vector. This vector preserves both the content information and citation information (network structure). The proposed model learns hidden latent features using the graph autoencoder structure. In the second module, the graph embeddings and citation history will be used to provide a ranked list of recommendations.

### 6.2.1 Problem Definition

In the previous approach, both the papers and the authors were considered as vertices in the graph. In this approach, only the papers are represented as vertices, while we use the authors as features. For that, we redefine the problem as follows: Given a bibliography dataset  $P$ , the corresponding bibliographic citation network is  $G = \langle V, E, A, C \rangle$ , where  $G$  is a directed graph and  $V$  the paper vertices set,  $V = \{v_i\}$  ( $1 \leq i \leq n$ ,  $n$  is the total number of papers).  $E$  is a set of edges representing the citation relation between the vertices.  $A$  is the adjacency matrix of graph  $G$  representing the network structure, where  $A_{i,j} = 1$  if  $e_{i,j} \in E$ .  $C$  is a matrix representing the vertices feature, where  $C_i \in C$  indicates the feature of vertex  $v_i$ . The potential vertex features for a vertex  $v_i$ , are the content information such as ( $\text{Doc2Vec}_{\text{title}_i}$ ,  $\text{Doc2Vec}_{\text{abstract}_i}$ ,  $\text{BOW}_{\text{title}_i}$ , and  $\text{BOW}_{\text{abstract}_i}$ ), and the citation information including ( $h_v$  and  $\text{authors}_i$ ). Where  $\text{title}_i$  is the title of vertex  $v_i$ , correspondingly  $\text{abstract}_i$  is the abstract and  $\text{authors}_i$  is the 1-hot vector representing the authors of paper  $i$ .  $h_i$  is a vector of all papers cited by the authors of  $v_i$ , where  $h_{i,j}$  is the number of times paper  $j$  was cited by  $\text{author}_i$ . Using the *Doc2Vec* model, documents with similar context will be close in the vector space. In the BOW model, a document is represented as a vector of words where weighting techniques like TF-IDF reflects the importance of a term with regard to the document in hand.

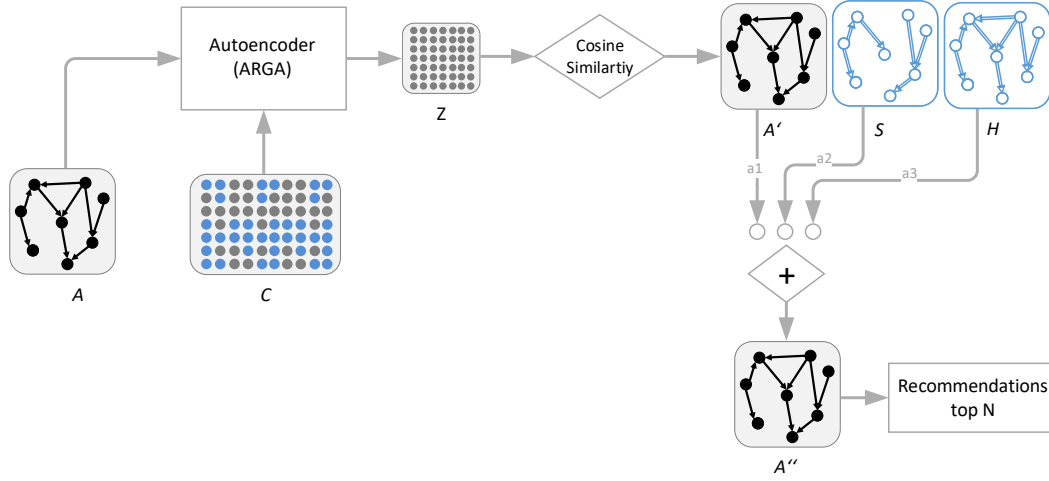


Figure 40: Block diagram of the proposed citation recommendation system.  $A$  is the graph structure, and  $C$  is the vertex content matrix.  $Z$  is the graph embeddings matrix generated using the Autoencoder (ARGA) model.  $A'$  is the adjacency matrix reconstructed from  $Z$ .  $S$  is the similarity matrix and  $H$  is the history of previously cited paper by the authors.  $A''$  is final adjacency matrix constructed by fusing  $A'$ ,  $S$ , and  $H$ .

### 6.2.2 Methodology

The structure of the proposed model is shown in Figure 40. The model consists of two main modules: the graph embeddings module and the recommendation module. Based on the graph embeddings module, we can learn an effective feature representation vector that preserves both content information and citation information. Given the network  $G$ , our graph embedding module learns a low-dimensional vector  $Z_i \in \mathbb{R}^d$  with the format as follows:  $f : (A, C) \mapsto Z$ , where  $Z_i$  is the embedding of the  $i_{th}$  row of the matrix  $Z \in \mathbb{R}^{N \times d}$ .  $Z$  is the embeddings matrix that encodes the structure and content information of the citation network.  $N$  is the number of vertices, and  $d$  is the dimension of embedding. We construct the generative adversarial network embedding model to obtain the graph embedding matrix  $Z$ .

Using the graph embeddings  $Z$  and the citation history  $H$ , we recommend a small subset of papers  $p \in P$  for a query manuscript  $q$ . In the following, we explain both modules in detail.

#### Graph Embeddings Module

Figure 41 illustrates the graph embeddings model. The structure used in this work is based on ARGAs proposed by Pan et al. [122], however, different structures for graph convolution, generator  $G$ , discriminator  $D$  and autoencoder are used. Also, we investigate the different features, their representation, and the corresponding hyper-parameters.

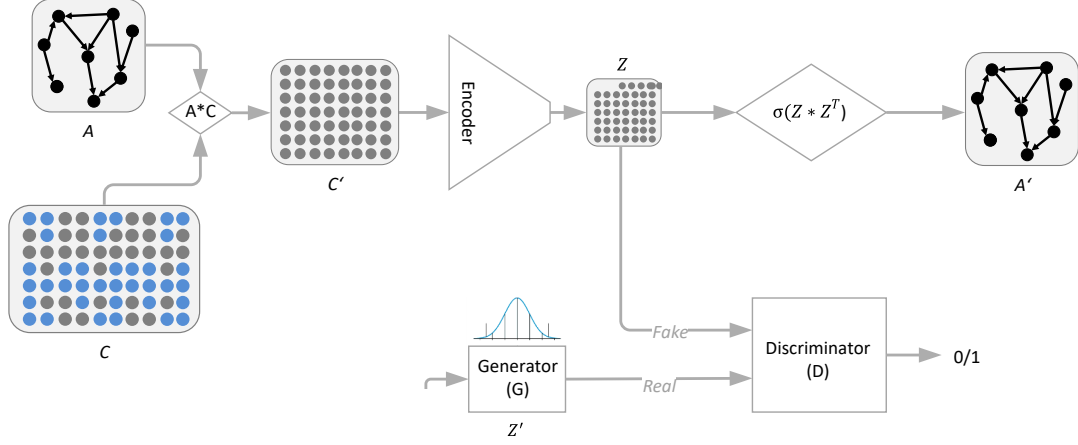


Figure 41: The architecture of the graph autoencoder (ARGA) model used to encode the graph structure  $A$ , and the vertex content matrix  $C$  into a low-dimensional space to learn a latent representation  $Z$ .  $A'$  is the adjacency matrix reconstructed from  $Z$ . The discriminator should predict whether a sample is generated from the graph embeddings or a prior distribution.

#### Graph Autoencoder (GAE):

The GAE encodes both the graph structure  $A$  and the vertex content (content information)  $C$  into a low-dimensional space to learn a latent representation  $Z$ .

The input to the model are a feature matrix  $C \in \mathbb{R}^{N \times m}$ , where  $m$  is the size of the feature vector and an adjacency matrix  $A \in \mathbb{R}^{N \times N}$  where  $N$  is the number of examples in the dataset. The autoencoder transforms the input to  $Z \in \mathbb{R}^{N \times d}$  in the encoder.

Using simple matrix multiplication  $C' = A * C$ , every paper inherits all the properties directly from the papers which it cites, but it does not keep its own. Hence, a paper with a lot of citations will have high, non-normalized scalars in its representation leading to unpredictable behaviour in further learning steps. To mediate this problem, we first add the identity matrix  $I$  to  $A$ , such that every paper will keep its own properties, and second we normalize the result. This is performed by calculating the weighting matrix  $D$ , where  $D_{i,i} = \text{row-sum}(A_i)$  and 0 everywhere else.

$$C' = D^{-1} \cdot (A + I) \cdot C \quad (22)$$

In order to account for the number of times, a paper is cited, we consider the approach by Kipf and Welling [88], where frequently cited papers have higher weight.

$$C' = D^{-\frac{1}{2}} \cdot (A + I) \cdot D^{-\frac{1}{2}} \cdot C \quad (23)$$

The encoder consists of multiple layers, reducing the dimensionality of  $C'$  to form the graph embeddings matrix  $Z$ . The goal is to match similar papers close to each other in the vector space. The decoder tries to reconstruct  $A$ , but it is merely a simple matrix multiplication of the embeddings:  $Z \times Z^T$ . The entries of  $A'$  are real numbers, and a high value at position  $(x, y)$  in  $A'$  is interpreted as a high similarity between papers  $p_i$  and  $p_j$ , where the  $i$ -th and the  $j$ -th rows are the feature representations in  $C$  of  $p_i$  and

$p_j$ . The goal is to match  $A'$  to  $A$  as close as possible. One could already recommend the citations at this point, but the history has not been considered yet. The adjacency matrix  $A$  is reconstructed by applying a sigmoid function  $A' = \sigma(Z \times Z^T)$ . This process can be performed by either a 'normal' autoencoder or a variational autoencoder.

#### *Generative Adversarial Network (GAN) Module:*

The adversarial model forces the latent representation  $Z$  to follow a prior distribution using an adversarial training model. The model consists of a generator  $G$  and a discriminator  $D$ . The generator outputs a prior distribution  $Z'$ . The discriminator of the GAN is connected to the 'real' embeddings  $Z$  and the generator output  $Z'$ , and discriminates whether  $z_i \in Z$  is coming from the autoencoder or from the generator. Just like in the adversarial autoencoder, the posterior distribution  $Z$  is thereby forced to follow the prior distribution  $Z'$ .

The graph embeddings model will be used to generate the graph embeddings corresponding to the query  $q$  and the bibliography dataset  $P$ . By calculating the relevance of the papers to the query using the cosine similarity of the graph embeddings, we can build the adjacency matrix  $A''$ .  $A''$  can be directly used for providing the recommendations.

#### *Citation Recommendation Module*

Authors tend to recite papers that they previously cited. In order to account for this pattern, we use the history matrix  $H$ .  $H$  is represented as a square matrix  $H \in \mathbb{N}^{N \times N}$ , where  $h_{i,j}$  is the count of times a paper  $j$  is cited in papers written by the authors  $i$  before; 0 otherwise. By defining multiple time intervals  $k$ , we can use the corresponding history matrices  $H = \{H_1, \dots, H_k\}$  to give importance to previously cited papers. In order to account for the implicit semantic similarity between papers, we use a square matrix  $S \in \mathbb{N}^{N \times N}$ , where  $S_{i,j} = \cos(\text{Doc2Vec}_{\text{abstract}_i}, \text{Doc2Vec}_{\text{abstract}_j})$  is the semantic similarity between the papers  $i, j$ , measured using the cosine similarity between their Doc2Vec representation.

Using the embeddings matrix  $Z$ , we rebuild  $A' : A'_{i,j} = \cos(z_i, z_j)$ .  $A'$  represents the cosine similarity between the different papers with regard to content and citation information. We construct matrix  $A'' = \alpha_1 A' + \alpha_2 S + \alpha_3 H_1 + \dots + \alpha_{k+1} H_k$ , where  $k$  is the number of time intervals and  $\alpha_x \in \mathbb{R}$ . The coefficients  $\alpha_x$  are optimized based on minimizing the mean square error between  $A$  and  $A''$ .

Finally, given a query  $q$  and the graph embeddings matrix  $Z$ , we recommend a ranked list of papers based on their similarity to the query embedding and the history referring to previously cited papers by the authors of  $q$ .

### 6.2.3 Dataset and Evaluation Settings

In order to evaluate the quality of the proposed model, we conduct experiments on two bibliographic datasets, namely AAN and DBLP. For the AAN dataset, the same training and test split from Section 6.1.5 were used.

DBLP consists of bibliography data in computer science [155]. Similarly to the approaches in related work, instead of using the full dataset, we chose a subset to avoid samples with incomplete references. We also removed papers that have missed titles or abstracts in the dataset. For ANN, we selected papers published before 2013 as the training set (11197 papers). We used papers published in 2012 as the validation set, and papers published from 2013 as the test set (1358 papers).

For DBLP, similarly to our baseline [17], we used a subset of the DBLP papers from conferences in five research areas: *Computational Linguistics & Information Retrieval* (SIGIR, PAKDD, CIKM, EMNLP, ECIR, NAACL, COLING, ACL, EACL), *Machine Learning* (ICML, SIGKDD, NIPS, ICDE, ICDM, WSDM), *Computer Vision* (CVPR, ECCV, ICIP, ICPR, MM, ICCV, ACCV), *Communication Networks* (MOBICOM, ICDCS, SECON, GLOBECOM, and ICNP, INFOCOM, SIGCOMM, ICC), and *Computer Security* (ACSAC, ARES, SP, NDSS, FC, ISI). We selected papers published until 2013 (included) as training set (56,304 papers). Similarly, we used papers published in 2012 as the validation set, and papers published from 2014 to 2015 as test set (8,028 papers).

The experiments carried out aim at studying the effectiveness of using the proposed model for personalized citation recommendation. We first analyze three different aspects, namely the network structure, the features used to represent a paper, and the auto-encoder structure used for embeddings creation. Based on the findings of these aspects, we define a system configuration that is used afterward to evaluate our approach against state-of-the-art techniques. For document representation, *Doc2Vec* model is trained using Wikipedia dump. The BOW model is built using the corresponding corpus, namely AAN and DBLP.

### 6.2.4 Model Structure Optimization

Generative adversarial learning models often fail to converge and are very sensitive to hyper-parameter initialization and optimization. In the following experiments we analyse the different parameters based on the performance of the graph embeddings model for providing citations. All the evaluations are performed on the AAN Dataset unless it is explicitly declared.

Each experiment analyses one aspect. For this purpose we fix the other hyperparameters. The default parameters are as follows: Each model is trained for 100 epochs with a batch size of 64 and a learning rate of 0.002. The dropout probability is fixed to 0.1, Adam-optimizer is used as an optimizer and =ReLU&Linear is used as the activation function. In ReLU&Linear, all layers use ReLU as activation function, except for the last layer connected to the final embeddings which is linear. These parameters were selected based on a greedy search through a specified subset of the space of these hyperparameters space. We use the BOW representation of the title and we represent

corresponding authors as 1-hot vector. We use an encoder model with three layers (H1,H2,H3)=(400, 200, 400).

#### *Autoencoder vs Variational Autoencoder*

The adversarially regularized graph autoencoder can use an autoencoder (AE) or a variational autoencoder (VAE). Table 29 shows that VAE outperforms AE in three categories while AE performs better in terms of Recall@100. This is due to the design of the VAE, where a *Kullback-Leibler divergence* is used as a penalty function to match the posterior distribution to a normal distribution  $N(0, 1)$  in addition to the prior distribution of the generator. For the final comparison with the approaches from related work, we consider VAE.

Table 29: Performance using autoencoder (AE) vs variational autoencoder (VAE).

<b>Model</b>	<b>MAP</b>	<b>MRR</b>	<b>Recall@20</b>	<b>Recall@100</b>
AE	0.047	0.168	0.092	0.376
VAE	0.053	0.233	0.106	0.256

#### *Prior Distribution*

By design, the generator forces the encoder to match its posterior distribution of the data to the prior distribution  $N(0, 1)$ . We evaluate the model performance using normal distribution  $N(\mu, \sigma)$  and uniform distribution  $U(\mu, \sigma)$ .

Table 30: The influence of the prior distribution on the performance of the recommender system.

<b>Layers:</b>	<b>MAP</b>	<b>MRR</b>	<b>Recall@20</b>	<b>Recall@100</b>
U(0, 1)	0.077	0.283	0.158	0.388
U(0, 3)	0.074	0.285	0.150	0.370
N(0, 1)	0.077	0.283	0.155	0.385
N(0, 3)	0.074	0.278	0.152	0.380
N(0, 0.05)	0.071	0.271	0.143	0.361
N(0, 0.13)	0.081	0.284	0.159	0.403

Table 30 shows that the embeddings with normal distribution ( $\mu = 0$  and  $\sigma = 0.1333$ ) performed better than the  $N(0, 1)$  distribution. From there, we can conclude that enforcing the embeddings to follow a normal distribution can improve the performance. In addition, it is better to optimize the structure based on analysing different prior distribution ranges than using a predefined distribution.

### Number of Layers

Table 31 shows that the encoder model with 3-layers gives the best performance. A model with deeper than 2 layers leads to better performance except for the Recall@20. Table 32 shows a different pattern for the discriminator, where the model with 2-layers gives the best performance. For that, we use an encoder with three hidden layers of 400 – 200 – 100 and a discriminator with three layers of 100 – 200 – 1 for the comparison against the baseline.

Table 31: The influence of the encoder depth on the performance of the recommender system.

Layers:	MAP	MRR	Recall@20	Recall@100
100	0.026	0.148	0.054	0.165
200-100	0.073	0.283	0.155	0.385
400-200-100	0.079	0.284	0.150	0.419

Table 32: The influence of the discriminator depth on the performance of the recommender system.

Layers:	MAP	MRR	Recall@20	Recall@100
100-1	0.062	0.255	0.127	0.318
100-200-1	0.077	0.282	0.155	0.385
100-200-200-1	0.074	0.281	0.1517	0.371

### Number of Neurons

In this step, we investigate how the number of hidden neurons affect the model performance. Table 33 shows that increasing the number of hidden neurons increases the performance of the model.

Table 33: The influence of the number of neurons on the performance of the recommender system.

$H_1, H_2, H_3$	MAP	MRR	Recall@20	Recall@100
32, 16, 32	0.024	0.132	0.055	0.157
64, 32, 64	0.041	0.191	0.086	0.258
100, 50, 100	0.057	0.236	0.117	0.319
200, 100, 200	0.077	0.282	0.155	0.385
400, 200, 400	0.076	0.279	0.151	0.378
600, 300, 600	0.081	0.290	0.161	0.410

### Activation Functions

The activation function of deep neural networks has a crucial impact on the performance of the training procedure. Their non-linearity makes it possible for the neural networks to learn more complex patterns. We will look at 4 different activation functions, namely ReLU, Tanh, Sigmoid and ReLU&Linear. In ReLU&Linear, all neurons use ReLU as activation function. Only the last layer connected to the final embeddings will be linear.

Table 34: The influence of activation functions on the performance of the recommender system.

Function	MAP	MRR	Recall@20	Recall@100
Tanh	0.112	0.315	0.239	0.529
ReLU&linear	0.076	0.279	0.151	0.378
ReLU	0.041	0.149	0.082	0.342
Sigmoid	0.003	0.027	0.006	0.017

Table 34 shows that the Tanh activation function clearly has better performance than the second-best ReLU&Linear. ReLU and Sigmoid functions hardly let the model learn. This can be justified by the fact that, ReLU&Linear and Tanh can map to negative values while ReLU maps to positive values between  $(0, \infty)$ . Hence, Tanh and ReLU&Linear have the ability to map to negative values in the vector space and therefore perform better. Sigmoid has led to the worst results as all of its vectors are crowded in a small vector space, making distinctions hardly possible.

### Vertex Features Analysis

The potential features for a vertex  $v_i$  are, namely  $\text{Doc2Vec}_{\text{title}_i}$ ,  $\text{Doc2Vec}_{\text{abstract}_i}$ ,  $\text{BOW}_{\text{title}_i}$ ,  $\text{BOW}_{\text{abstract}_i}$ ,  $h_v$ ,  $\text{authors}_i$ . In the following experiment, we try different combinations of features.

Table 35: The influence of the feature set on the performance of the recommender system.

Node Features	MAP	MRR	Recall@20	Recall@100
$\text{BOW}_{\text{title}_i}, \text{authors}_i$	0.076	0.279	0.151	0.378
$\text{BOW}_{\text{title}_i}, \text{BOW}_{\text{abstract}_i}, \text{authors}_i$	0.078	0.270	0.165	0.401
$\text{BOW}_{\text{title}_i}, \text{BOW}_{\text{abstract}_i}, \text{authors}_i, H_i$	0.080	0.275	0.168	0.409
$\text{Doc2Vec}_{\text{title}_i}, \text{Doc2Vec}_{\text{abstract}_i}, H_i$	0.080	0.278	0.162	0.367

Table 35 indicates that the best performance is achieved with the 3rd combination, which uses the abstract, title, and history. Feeding history as a feature will increase the complexity of feature space. For that, we will use the second combination which takes only the co-authorship and content into consideration since the performance is relatively similar.



Table 36: Comparison with the baselines on AAN dataset.

Approach	MAP	MRR	Recall@20	Recall@40	Recall@100
HITS [120]	0.073	0.089	0.186	0.258	0.378
MultiLayer Graph Model [121]	0.091	0.108	0.209	0.275	0.396
Unified Graph Model [110]	0.108	0.117	0.221	0.286	0.410
LocDiSCern [19]	0.113	0.120	0.226	0.292	0.421
GloDiSCern [19]	0.112	0.119	0.225	0.290	0.418
Neural Probabilistic Model [74]	0.119	0.126	0.234	0.299	0.441
Mutually Reinforced Model [17]	0.126	0.137	0.242	0.331	0.479
Our Approach	<u>0.148</u>	<u>0.399</u>	<u>0.290</u>	<u>0.419</u>	<u>0.585</u>

Table 37: Comparison with the baselines on DBLP dataset.

Approach	MAP	MRR	Recall@20	Recall@40	Recall@100
HITS [120]	0.051	0.063	0.159	0.232	0.350
MultiLayer Graph Model [121]	0.007	0.082	0.181	0.253	0.374
Unified Graph Model [110]	0.058	0.098	0.201	0.266	0.387
LocDiSCern [19]	0.092	0.107	0.214	0.275	0.395
GloDiSCern [19]	0.091	0.105	0.212	0.273	0.392
Neural Probabilistic Model [74]	0.099	0.113	0.221	0.282	0.402
Mutually Reinforced Model [17]	0.115	0.125	0.228	0.302	<u>0.428</u>
Our Approach	<u>0.152</u>	<u>0.222</u>	<u>0.319</u>	<u>0.363</u>	0.405

#### 6.2.5 Comparison with other Personalized Citation Recommendation Systems

We compare our approach with other state-of-the-art personalized recommendation systems using the ANN and DBLP datasets and following the same split for training and testing the models. The hyperparameters are selected based on the previous analysis and only one matrix  $h_1$  for the history is considered. The experiments show that defining multiple intervals did not result in significant improvement. The compared approaches are the same as in Section 6.1.8. Table 36 shows that the proposed model outperforms the other approaches with a wide margin on the AAN. While Table 37 shows that the proposed model provides comparable performance to existing approaches on the DBLP dataset. Which indicates a better ranking performance of the proposed model.

Table 38 shows that the reconstructed adjacency matrix  $A'$  has more impact compared to the semantic similarity matrix  $S$  and the history of previously cited papers  $H$ . This insight is significant since it shows that the compact graph embedding can capture both similarities and interactions between the papers and implicitly account for the history of previously cited papers.

Table 38: The coefficients of different feature matrices on the performance of the recommender system.

Dataset	A'	Cosine	History
AAN	0.996	0.184	0.015
DBLP	1.074	0.074	-0.041

### 6.3 DISCUSSION AND OWN CONTRIBUTIONS

In this chapter, we addressed the challenge of combining different heterogeneous information sources for personalized citation recommendation. For that, we have proposed two methods that leverage citation and content information. The first method incorporates two modules, namely a query-based recommendation module and a graph-based ranking module. The query-based recommendation module uses an unsupervised DSSM model to rank papers based on their semantic similarity to the query text. The DSSM proved a better performance compared to a basic content-based recommender system that uses Doc2Vec to represent the query text and the candidate documents. The graph-based ranking builds a multi-layer graph of papers and their citations to rank papers based on both citation information and content information. The final list of recommendations is formed by averaging the ranks of the recommendations provided by both modules. Our evaluation using the AAN dataset proved that the proposed model outperforms the baseline techniques with a wide margin.

The main advantage of this method is that integrating new papers in the system requires only an update of the multi-layered graph. Another advantage is the flexibility to add other types of information to the graph ranking module to account for other aspects, i.e., venue and year of publication. However, it ignores the latent similarity between authors. This refers to papers of authors with similar citation and content information which will not be recommended unless they were cited before by the query authors.

The second approach addressed this limitation by using an ARGA model to project the citation and content information into a low-dimensional and compact feature space. The proposed model learns hidden latent features using the graph autoencoder to represent the heterogeneous bibliographic information. Then, the graph embeddings and the history of cited papers previously by the authors were used to provide the final list of recommendations. Our evaluation of the proposed approach using the AAN and DBLP datasets proved the superior performance of the proposed model compared with other personalized recommendation systems. The main limitation of this model, is that the recommendation system should be retrained in order to account for new papers and authors.



## SUMMARY AND OUTLOOK

---

To conclude this thesis, the content of the previous chapters and the main contributions are summarized. We then draw conclusions and give an outlook of future research opportunities.

### 7.1 SUMMARY OF THESIS

In Chapter 1, we motivated the benefit of using the rich knowledge embedded in unstructured documents and lexical-semantic resources as well as deep learning structures to address research challenges in three major blocks necessary to build linked data platforms, namely ontology construction and enrichment, multi-label text classification and personalized citation recommendation. Following the motivation and challenges, we described in detail the research goals related to the aforementioned tasks. Based on the background knowledge presented in Chapter 2 and our analysis of the related work relevant to our contributions in Chapter 3, we presented the following contributions.

#### 7.1.1 Contributions

The first contribution of this work focused on using existing lexical-semantic resources, deep learning structures, and word embeddings to minimize the intensive reliance on complicated feature engineering and linguistic analysis for ontology construction and enrichment. In Chapter 4, we described Onto.KOM as a minimally supervised framework for ontology learning. First, we showed that ontological categories can be automatically extracted by grouping words based on the cosine similarity of the corresponding embeddings. Word embeddings preserve linguistic regularities, such as word similarity and analogy. For that, we evaluated three different approaches for ontology enrichment using the embeddings of word-pairs. We showed that enriching ontologies with new relations based on the embedding offset is more complex than reliance on similarity scores between words. Motivated by these results, we extended Onto.KOM to support multi-way semantic relation classification. We closely analysed the influence of three aspects on the classification performance: the word embeddings used, the relation representation, and the CNN structure. Further on, we investigated the effectiveness of using Onto.KOM to classify word-pairs over multiple semantic relations and the learning capability of the network with an increased number of training samples. Onto.KOM significantly outperforms two state-of-the-art techniques for semantic relation classification, namely *Taxi*[124], *Att-Input-CNN*[162]. Our approach achieved an accuracy of 68.6% compared to 24.9% and 57.5% for *Taxi*, and *Att-Input-CNN* respectively. We showed that using lexical-semantic resources,

deep learning, and word embeddings can reduce handcrafted feature engineering in previous techniques for ontology learning. Research objective of  $RG_1$  was thus achieved.

In the scope of  $RG_2$ , we proposed models to overcome the challenges of the high dimensionality of feature space, label imbalance, and training overhead in multi-label text classification. In our second contribution, we proposed a new method to select semantic-based features using only the typed dependencies without relying on any external lexical databases, dictionaries, or syntactic patterns in Section 5.2. We improved on our previous work using taxonomic relations by relying on the typed dependencies, which can identify these shallow relations even with more natural texts since they analyse syntactic relations on the sentence level. Our comprehensive evaluation against a wide range of feature selection techniques proved that taking the syntactic and semantic relations between words into consideration can provide better performance concerning the compared statistical and semantic-based approaches. In addition, our model significantly reduces the computation costs for selecting features by relaying on the built ontology for selecting and updating the features. We are thus making a contribution to  $RG_{2.1}$ .

The third contribution in this work focused on exploring how deep learning structures and distributed representations can address the main challenges of traditional classifiers with regards to the used feature selection technique and the trained classifier. We address the case of a small dataset of long documents, where traditional classifiers tend to perform better than deep learning structures. In our experiments, we showed that the tedious process of feature selection and classifier selection can be eliminated by using the distributed representations of document fragments and deep learning structures. Our best performing Stacked GRU using chunks with Doc2Vec representations is lightweight and quickly trained in comparison to models utilizing words as input. Our approach outperformed the baselines on two datasets, namely EUR-Lex and RCV1 in terms of  $F1_{micro}$  and  $F1_{macro}$ . Hence, we provided contributions to  $RG_{2.2}$ .

The fourth contribution targets the label-imbalance problem. We presented an ontology-based training-less classifier, which solely relies on developing a domain ontology to be used as a training-less classifier without a pre-classified set of training documents. Thereby, we overcome the challenges of feature engineering and label imbalance in traditional methods. The built model is not affected by the label imbalance problem. Besides, updating the labelset requires only updating the label ontologies, which maximizes the system scalability. Our intensive experiments proved that our method can provide a fair performance for the different labels. The performance of our approach is comparable to other approaches, but it requires no training. By that, the research objective of  $RG_{2.3}$  was achieved.

To achieve the research objectives of  $RG_3$ , we presented two approaches for providing personalized citation recommendations by leveraging the different heterogeneous information sources corresponding to citation and content information. The fifth contribution focused on using an ensemble model of two modules, namely a query-based recommendation module and a graph-based ranking module. The query-based recom-

mentation module ranks papers based on their semantic similarity to the query text. The graph-based ranking module ranks papers based on an algorithm that traverses a graph representing authors, papers, citation information, and content information of all candidate papers. Averaging the ranks of the recommendations provided by both modules forms the final list of recommendations. Our evaluation on the AAN dataset proved that the proposed model significantly outperforms the baseline methods. Using the proposed system, global and local personalized citation recommendations can be realized by using the title, abstract, full manuscript, or only short text corresponding to a specific context respectively. Other semantic representations of candidate papers and the query text, instead of word hashing, might improve the model performance. It is also interesting to investigate the influence of other information sources, i.e., the venue information. The main advantage of the first method is that integrating new papers in the system requires only updating the multi-layered graph, however, it ignores the latent similarity between authors.

The final contribution focused on using adversarially regularized graph autoencoders to learn effective graph embeddings that preserve both citation and content information. The proposed model learns hidden latent features using the graph autoencoder to represent the heterogeneous bibliographic information. Then, the graph embeddings and the history of previously cited papers by the authors were used to provide the final list of recommendations. Our evaluation of the proposed approach on the AAN and DBLP datasets showed that encoding the citation and content information into a lower-dimensional space can improve the performance of personalized citation recommendation. This model showed a better performance compared to the first method. However, the recommendation system should be retrained in order to account for new papers. Overall, the performance of both approaches proved our contributions to RG<sub>3</sub>.

## 7.2 OUTLOOK

The work presented provides the foundation to realize linked-data platforms from unstructured content in academia and OER platforms to make resources more accessible and discoverable. Our contributions pave the road for further research in the field of enriching information sources with semantic metadata by leveraging the wealth of information embedded in lexical-semantic resources and unstructured data. In addition, follow-up research can be pursued based on our results concerning using deep learning and distributed representations to combine heterogeneous information sources to provide better personalized recommendations.

With regards to our approach for ontology construction and enrichment Onto.KOM, there is a potential to integrate additional lexical-semantic resources to improve the performance as well as covering more types of relations. While the method used in Onto.KOM is very general and easy to use; it might be better in specific use-cases to augment the input with additional features. For example, the entire input sentence or other contextual data such as the author of a piece of text or the date on which it was written could be used. Also, contextualized word embeddings can be used to provide a

better representation of words in specific domains. Evaluating the difference between these alternatives could further verify the hypothesis that the CNN classifier can infer some of the context on its own. Even though we only extracted taxonomic relations using the lexico-syntactic pattern, it is also possible to crawl other types of relations using bootstrapping-based statistical and linguistics methods. Finally, Onto.KOM can be used as a stand-alone system or as a complementary component of other ontology learning systems to evaluate the quality of extracted relations based on previously observed ones.

Concerning our models for multi-label text classification, new datasets can be used to analyse the predictive performance of our methods under different dataset properties. Each of the proposed approaches addresses different research gaps. The proposed model for feature selection using typed dependencies can be extended by leveraging the rich information embedded in lexical-semantic resources databases. Deep learning for NLP is a very active research field, new contextualized word embeddings such as Flair [1], and ELMO [128] as well as new structures like *Transformers* [37] can be analysed in the context of small datasets of long documents. Concerning our proposed training-less classifier, more work can be done in the area of building the domain ontology by integrating more lexical resources. An ensemble approach of the proposed models can be used to leverage the advantages of the different models by using deep learning approaches for frequent labels and the training-less classifier for the less frequent labels.

In our current approach for personalized citation recommendation using graph embeddings, we used only the content and authors' information as node features. It would be interesting to study the influence of other features, i.e., venue and publication year on the quality of generated graph embeddings. The assumption is that the authors tend to cite more recent papers published in reputed venues. To stimulate critical thinking and counter biases, recommendations should differ from those that an author already knows. Overall, recommendation systems should therefore provide novel, diverse and serendipitous recommendations. There is a potential of using lexical-semantic resources and distributed representations to address these three aspects by manipulating the user query to include related concepts.

To demonstrate the eligibility of our contributions, the proposed approaches have been applied for personalized books recommendation in the scope of the *Personalized Interest-oriented Book Recommendation (PIOBRec)* project as part of the Software Campus program funded by the German Federal Ministry of Education and Research (BMBF)<sup>1</sup>. The project was carried out in collaboration with three German trade publishers of *Holtzbrinck*. The publishers want to utilize recommender systems to maximize the user's satisfaction and thus increase their return of investment (ROI). The key challenge lies within the lack of metadata representing their large collection of books. Around 20% of the books are indexed using a categorization scheme. The assignment of categories is done manually by editors, which is time-consuming and very expensive. The categorization scheme covers multiple aspects, such as genre, location, and

---

<sup>1</sup> <https://softwarecampus.de/>

time period. In addition, it is imbalanced. Using our Stacked GRU we train a classifier for the frequent labels in the categorization scheme. We exploit the training-less classifier for categorizing books using the less frequent labels. Using word embeddings, we enriched the books with additional metadata by adding similar words. For example, if we have *Fiktion* then we add *Belletristik* to the extracted metadata of a book. The publishers have sales information about their books, such as Bought-together, and Also-viewed. The challenge is how to combine the sales information with content information to provide better recommendations. Using our ensemble model, we proved that better recommendations could be delivered. In general, each of the presented contributions exposes the potential to design and develop new approaches that can build upon the methods, systems, and results presented in this work.





## BIBLIOGRAPHY

---

- [1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. "Contextual string embeddings for sequence labeling." In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 1638–1649.
- [2] Wael Alkhatib, Leon Alexander Herrmann, and Christoph Rensing. "Onto.KOM - Towards a Minimally Supervised Ontology Learning System based on Word Embeddings and Convolutional Neural Networks." In: *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 2017, pp. 17–26.
- [3] Wael Alkhatib, Christoph Rensing, and Johannes Silberbauer. "Multi-label text classification using semantic features and dimensionality reduction with autoencoders." In: *International Conference on Language, Data and Knowledge*. Springer. 2017, pp. 380–394.
- [4] Wael Alkhatib, Saba Sabrin, Svenja Neitzel, and Christoph Rensing. "Towards Ontology-Based Training-Less Multi-label Text Classification." In: *International Conference on Applications of Natural Language to Information Systems*. Springer. 2018, pp. 389–396.
- [5] Wael Alkhatib, Steffen Schnitzer, and Christoph Rensing. "Training-Less Multi-label Text Classification Using Knowledge Bases and Word Embeddings." In: *International Conference on Knowledge Science, Engineering and Management*. Springer. 2019, pp. 97–104.
- [6] Chidanand Apté, Fred Damerau, and Sholom M Weiss. "Automated learning of decision rules for text categorization." In: *ACM Transactions on Information Systems (TOIS)* 12.3 (1994), pp. 233–251.
- [7] Mark J Berger. *Large Scale Multi-label Text Classification with Semantic Word Vectors*. Technical Report. 2015.
- [8] Matthew Berland and Eugene Charniak. "Finding parts in very large corpora." In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics. 1999, pp. 57–64.
- [9] Steven Bethard and Dan Jurafsky. "Who should I cite: learning literature search models from citation behavior." In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM. 2010, pp. 609–618.
- [10] Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. "The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics." In: *Proceedings of the Sixth International Conference*

- on Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA), May 2008.
- [11] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information." In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.
  - [12] Georgeta Bordea, Els Lefever, and Paul Buitelaar. "Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2)." In: *SemEval-2016*. Association for Computational Linguistics. 2016, pp. 1081–1091.
  - [13] Willem Nico Borst. *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente, 1997.
  - [14] Elizabeth Boschee, Ralph Weischedel, and Alex Zamanian. "Automatic information extraction." In: *Proceedings of the International Conference on Intelligence Analysis*. Vol. 71. Citeseer. 2005.
  - [15] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. "Ontology learning from text: An overview." In: *Ontology learning from text: Methods, evaluation and applications* 123 (2005), pp. 3–12.
  - [16] Robin Burke. "Knowledge-based recommender systems." In: *Encyclopedia of library and information systems* 69. Supplement 32 (2000), pp. 175–186.
  - [17] Xiaoyan Cai, Junwei Han, Wenjie Li, Renxian Zhang, Shirui Pan, and Libin Yang. "A three-layered mutually reinforced model for personalized citation recommendation." In: *IEEE transactions on neural networks and learning systems* 29.12 (2018), pp. 6026–6037.
  - [18] Xiaoyan Cai, Junwei Han, and Libin Yang. "Generative Adversarial Network Based Heterogeneous Bibliographic Network Representation for Personalized Citation Recommendation." In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 5747–5754.
  - [19] Tanmoy Chakraborty, Natwar Modani, Ramasuri Narayanam, and Seema Nagar. "Discern: a diversified citation recommendation system for scientific queries." In: *2015 IEEE 31st international conference on data engineering*. IEEE. 2015, pp. 555–566.
  - [20] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. "Extreme Multi-Label Legal Text Classification: A case study in EU Legislation." In: *CoRR abs/1905.10892* (2019). arXiv: 1905.10892.
  - [21] Kannan Chandrasekaran, Susan Gauch, Praveen Lakkaraju, and Hiep Phuc Luong. "Concept-based document recommendations for citeseer authors." In: *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer. 2008, pp. 83–92.

- [22] Yi-Hsing Chang and Hsiu-Yi Huang. "An automatic document classifier system based on naive bayes classifier and ontology." In: *Machine Learning and Cybernetics, 2008 International Conference on*. Vol. 6. IEEE. 2008, pp. 3144–3149.
- [23] Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. "Ensemble application of convolutional and recurrent neural networks for multi-label text categorization." In: *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE. 2017, pp. 2377–2383.
- [24] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1724–1734.
- [25] Wei-Ta Chu and Ya-Lun Tsai. "A hybrid recommendation system considering visual information for predicting favorite restaurants." In: *World Wide Web* 20.6 (2017), pp. 1313–1331.
- [26] Stephanie Chua and Narayanan Kulathuramaiyer. "Feature Selection Based on Semantics." In: *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*. Springer, 2008, pp. 471–476.
- [27] Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Saric, and Isabel Rojas. "Unsupervised learning of semantic relations between concepts of a molecular biology ontology." In: *IJCAI*. Citeseer. 2005, pp. 659–664.
- [28] Amanda Clare and Ross D King. "Knowledge discovery in multi-label phenotype data." In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2001, pp. 42–53.
- [29] Guillaume Cleuziou and Jose G Moreno. "Qassit at semeval-2016 task 13: On the integration of semantic vectors in pretopological spaces for lexical taxonomy acquisition." In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. 2016, pp. 1315–1319.
- [30] Francesco Colace, Massimo De Santo, Luca Greco, Flora Amato, Vincenzo Moscato, and Antonio Picariello. "Terminological ontology learning and population using latent dirichlet allocation." In: *Journal of Visual Languages & Computing* 25.6 (2014), pp. 818–826.
- [31] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. "Natural language processing (almost) from scratch." In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2493–2537.
- [32] Ann Copestake. *Introduction to Natural Language Processing*. 2012.
- [33] Claudia D'Amato, Steffen Staab, A. Tettamanzi, T. Minh, and Fabien Gandon. "Ontology Enrichment by Discovering Multi-Relational Association Rules from Ontological Knowledge Bases." In: *Proc. of the ACM Int. Symposium on Applied Computing (SAC 2016)*. ACM, 2016.

- [34] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. "Adversarial network embedding." In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [35] Marie-Catherine De Marneffe and Christopher D Manning. *Stanford typed dependencies manual*. Tech. rep. Technical report, Stanford University, 2008.
- [36] Bernard Desgraupes. "Clustering indices." In: *University of Paris Ouest-Lab ModalX 1* (2013), p. 34.
- [37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).
- [38] Ying Ding and Robert Engels. "IR and AI: Using co-occurrence theory to generate lightweight ontologies." In: *12th International Workshop on Database and Expert Systems Applications*. IEEE. 2001, pp. 961–965.
- [39] Hang Dong, Wei Wang, Kaizhu Huang, and Frans Coenen. "Joint Multi-Label Attention Networks for Social Text Annotation." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 1348–1354.
- [40] Gauthier Doquire and Michel Verleysen. "Feature selection for multi-label classification problems." In: *International work-conference on artificial neural networks*. Springer. 2011, pp. 9–16.
- [41] Stephen Downes. "Models for sustainable open educational resources." In: *Interdisciplinary Journal of E-Learning and Learning Objects 3.1* (2007), pp. 29–44.
- [42] Hendrik Drachsler, Katrien Verbert, Olga C Santos, and Nikos Manouselis. "Panorama of recommender systems to support learning." In: *Recommender systems handbook*. Springer, 2015, pp. 421–451.
- [43] Jingcheng Du, Qingyu Chen, Yifan Peng, Yang Xiang, Cui Tao, and Zhiyong Lu. "ML-Net: multi-label classification of biomedical texts with deep neural networks." In: *arXiv preprint arXiv:1811.05475* (2018).
- [44] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. "Inductive Learning Algorithms and Representations for Text Categorization." In: *7th International Conference on Information and Knowledge Management*. Jan. 1998, pp. 148–152.
- [45] Travis Ebesu and Yi Fang. "Neural citation network for context-aware citation recommendation." In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 1093–1096.
- [46] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. "A multi-view deep learning approach for cross domain user modeling in recommendation systems." In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2015, pp. 278–288.

- [47] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.
- [48] David Faure and Claire Nedellec. "Knowledge acquisition of predicate argument structures from technical texts using machine learning: The system ASIUM." In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer. 1999, pp. 329–334.
- [49] Hermine Njike Fotzo and Patrick Gallinari. "Learning n-generalization/specialization relations between concepts: application for automatically building thematic document hierarchies." In: *Coupling approaches, coupling media and coupling languages for information retrieval*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE. 2004, pp. 143–155.
- [50] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. "Learning Semantic Hierarchies via Word Embeddings." In: *ACL (1)*. 2014, pp. 1199–1209.
- [51] Johannes Fürnkranz. "A study using n-gram features for text categorization." In: *Austrian Research Institute for Artificial Intelligence* 3.1998 (1998), pp. 1–10.
- [52] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Aistats*. Vol. 9. 2010, pp. 249–256.
- [53] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323.
- [54] Yoav Goldberg. "Neural network methods for natural language processing." In: *Synthesis Lectures on Human Language Technologies* 10.1 (2017), pp. 1–309.
- [55] Yuyun Gong and Qi Zhang. "Hashtag Recommendation Using Attention-Based Convolutional Neural Network." In: *IJCAI*. 2016, pp. 2782–2788.
- [56] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning, vol. 1*. 2016.
- [57] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.
- [58] Siddharth Gopal and Yiming Yang. "Multilabel classification with meta-level features." In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2010, pp. 315–322.
- [59] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks." In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2016, pp. 855–864.
- [60] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

- [61] Asela Gunawardana and Guy Shani. "A survey of accuracy evaluation metrics of recommendation tasks." In: *Journal of Machine Learning Research* 10.Dec (2009), pp. 2935–2962.
- [62] Ruining He and Julian McAuley. "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering." In: *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee. 2016, pp. 507–517.
- [63] Tingting He, Xiaopeng Zhang, and Xinghuo Ye. "An approach to automatically constructing domain ontology." In: *Proceedings of the 20th Pacific Asia conference on language, information and computation*. 2006, pp. 150–157.
- [64] Marti A Hearst. "Automatic acquisition of hyponyms from large text corpora." In: *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics. 1992, pp. 539–545.
- [65] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals." In: *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics. 2009, pp. 94–99.
- [66] Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. "A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model." In: *Knowledge-Based Systems* 97 (2016), pp. 188–202.
- [67] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. "Session-based recommendations with recurrent neural networks." In: *arXiv preprint arXiv:1511.06939* (2015).
- [68] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. "Recommending and evaluating choices in a virtual community of use." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/ Addison-Wesley Publishing Co. 1995, pp. 194–201.
- [69] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [70] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia." In: *Artificial Intelligence* 194 (2013), pp. 28–61.
- [71] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. "Knowledge-based weak supervision for information extraction of overlapping relations." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 541–550.

- [72] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. "Unbiased recursive partitioning: A conditional inference framework." In: *Journal of Computational and Graphical statistics* 15.3 (2006), pp. 651–674.
- [73] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. "Learning deep structured semantic models for web search using click-through data." In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM. 2013, pp. 2333–2338.
- [74] Wenyi Huang, Zhaohui Wu, Chen Liang, Prasenjit Mitra, and C Lee Giles. "A neural probabilistic model for context based citation recommendation." In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.
- [75] Xin Huang, Boli Chen, Lin Xiao, and Liping Jing. "Label-aware Document Representation via Hybrid Attention for Extreme Multi-Label Text Classification." In: *arXiv preprint arXiv:1905.10070* (2019).
- [76] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. "Label ranking by learning pairwise preferences." In: *Artificial Intelligence* 172.16-17 (2008), pp. 1897–1916.
- [77] Maciej Janik and Krys Kochut. "Training-less ontology-based text categorization." In: *Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR 2008) at the 30th European Conference on Information Retrieval, ECIR*. Vol. 20. 2008.
- [78] Maciej Janik and Krys J Kochut. "Wikipedia in action: Ontological knowledge in text categorization." In: *Semantic Computing, 2008 IEEE International Conference on*. IEEE. 2008, pp. 268–275.
- [79] Stanisaw Jastrzebski, Damian Leniak, and Wojciech Marian Czarnecki. "How to evaluate word embeddings? On importance of data efficiency and simple supervised tasks." In: *arXiv preprint arXiv:1702.02170* (2017).
- [80] Arif E Jinha. "Article 50 million: an estimate of the number of scholarly articles in existence." In: *Learned Publishing* 23.3 (2010), pp. 258–263.
- [81] Rob Johnson, Anthony Watkinson, and Michael Mabe. "The STM Report: An overview of scientific and scholarly publishing." In: *International Association of Scientific, Technical and Medical Publishers* (2018).
- [82] SungKu Kang, Lalit Patil, Arvind Rangarajan, Abha Moitra, Tao Jia, Dean Robinson, and Debasish Dutta. "Extraction of manufacturing rules from unstructured text using a semantic framework." In: *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection. 2015.
- [83] Zhao Kang, Chong Peng, and Qiang Cheng. "Top-n recommender system via matrix completion." In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.
- [84] KV Kanimozhi and M Venkatesan. "Unstructured data analysis-a survey." In: *International Journal of Advanced Research in Computer and Communication Engineering* 4.3 (2015), pp. 223–225.



- [85] Aurangzeb Khan, Baharum Baharudin, and Khairullah Khan. "Semantic based features selection and weighting method for text classification." In: *Information Technology (ITSim), 2010 International Symposium in*. Vol. 2. IEEE. 2010, pp. 850–855.
- [86] Han-Joon Kim and Ki-Joo Hong. "Building Semantic Concept Networks by Wikipedia-Based Formal Concept Analysis." In: *Advanced Science Letters* 21.3 (2015), pp. 435–438.
- [87] Yoon Kim. "Convolutional Neural Networks for Sentence Classification." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751.
- [88] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: *CoRR abs/1609.02907* (2016). arXiv: 1609.02907.
- [89] Thomas N. Kipf and Max Welling. *Variational Graph Auto-Encoders*. 2016. arXiv: 1611.07308 [stat.ML].
- [90] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents." In: *International Conference on Machine Learning*. 2014, pp. 1188–1196.
- [91] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. "DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia." In: *Semantic Web* 6.2 (2015), pp. 167–195.
- [92] Ladislav Lenc and Pavel Král. "Combination of Neural Networks for Multi-label Document Classification." In: *International Conference on Applications of Natural Language to Information Systems*. Springer. 2017, pp. 278–282.
- [93] David D Lewis. "Feature selection and feature extraction for text categorization." In: *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics. 1992, pp. 212–217.
- [94] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. "Rcv1: A new benchmark collection for text categorization research." In: *Journal of machine learning research* 5.Apr (2004), pp. 361–397.
- [95] Donald AB Lindberg, Betsy L Humphreys, and Alexa T McCray. "The unified medical language system." In: *Yearbook of Medical Informatics* 2.01 (1993), pp. 41–51.
- [96] ChunYang Liu, WenBo Sun, WenHan Chao, and Wanxiang Che. "Convolution neural network for relation extraction." In: *International Conference on Advanced Data Mining and Applications*. Springer. 2013, pp. 231–242.
- [97] Hugo Liu and Push Singh. "ConceptNeta practical commonsense reasoning tool-kit." In: *BT technology journal* 22.4 (2004), pp. 211–226.

- [98] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. "Deep learning for extreme multi-label text classification." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 115–124.
- [99] Ying Liu, Han Tong Loh, and Wen Feng Lu. "Deriving taxonomy from documents at sentence level." In: *HAD Prado and E. Ferneda, "Emerging Technologies of Text Mining: Techniques and Applications", Idea, Hershey, PA (2007)*, pp. 99–119.
- [100] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. "Yago3: A knowledge base from multilingual wikipeidias." In: *7th Biennial Conference on Innovative Data Systems Research*. CIDR Conference. 2014.
- [101] Promita Maitra and Dipankar Das. "JUNLP at SemEval-2016 Task 13: A language independent approach for hypernym identification." In: *Proceedings of SemEval (2016)*, pp. 1310–1314.
- [102] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. "Adversarial Autoencoders." In: *arXiv preprint arXiv:1511.05644 (2015)*.
- [103] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. "The stanford corenlp natural language processing toolkit." In: *ACL (System Demonstrations)*. 2014, pp. 55–60.
- [104] Takeshi Masuyama and Hiroshi Nakagawa. "Cascaded feature selection in SVMs text categorization." In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer. 2003, pp. 588–591.
- [105] Diana Maynard, Kalina Bontcheva, and Isabelle Augenstein. "Natural language processing for the semantic web." In: *Synthesis Lectures on the Semantic Web: Theory and Technology 6.2 (2016)*, pp. 1–194.
- [106] Sean M McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K Lam, Al Mamunur Rashid, Joseph A Konstan, and John Riedl. "On the recommending of citations for research papers." In: *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. ACM. 2002, pp. 116–125.
- [107] Sean Michael Mcnee. *Meeting user information needs in recommender systems*. University of Minnesota, 2006.
- [108] Eneldo Loza Menca and Johannes Furnkranz. "Pairwise learning of multilabel classifications with perceptrons." In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 2899–2906.
- [109] Eneldo Loza Menca and Johannes F rnkranz. "Efficient multilabel classification algorithms for large-scale problems in the legal domain." In: *Semantic Processing of Legal Texts*. Springer, 2010, pp. 192–215.
- [110] Fanqi Meng, Dehong Gao, Wenjie Li, Xu Sun, and Yuexian Hou. "A unified graph model for personalized query-oriented reference paper recommendation." In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM. 2013, pp. 1509–1512.

- [111] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.
- [112] George A Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41.
- [113] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. "Distant supervision for relation extraction without labeled data." In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics. 2009, pp. 1003–1011.
- [114] Hatem Mousselly Sergieh and Beto Boullosa. *Linguistic Analysis Levels*. 2016.
- [115] Cristiano Nascimento, Alberto HF Laender, Altigran S da Silva, and Marcos André Gonçalves. "A source independent framework for research paper recommendation." In: *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*. ACM. 2011, pp. 297–306.
- [116] Roberto Navigli and Simone Paolo Ponzetto. "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network." In: *Artificial Intelligence* 193 (2012), pp. 217–250.
- [117] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence  $O(1/k^2)$ ." In: *Doklady an SSSR*. Vol. 269. 3. 1983, pp. 543–547.
- [118] Thien Huu Nguyen and Ralph Grishman. "Relation extraction: Perspective from convolutional neural networks." In: *Proceedings of NAACL-HLT*. 2015, pp. 39–48.
- [119] Hiroshi Ogura, Hiromi Amano, and Masato Kondo. "Feature selection with a measure of deviations from Poisson in text categorization." In: *Expert Systems with Applications* 36.3 (2009), pp. 6826–6832.
- [120] Manabu Ohta, Toshihiro Hachiki, and Atsuhiko Takasu. "Related paper recommendation to support online-browsing of research papers." In: *Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011)*. IEEE. 2011, pp. 130–136.
- [121] Linlin Pan, Xinyu Dai, Shujian Huang, and Jiajun Chen. "Academic paper recommendation based on heterogeneous graph." In: *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Springer, 2015, pp. 381–392.
- [122] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. "Adversarially regularized graph autoencoder for graph embedding." In: *arXiv preprint arXiv:1802.04407* (2018).
- [123] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. "Tri-party deep network representation." In: *Network* 11.9 (2016), p. 12.

- [124] Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cédric Fairon, Simone Paolo Ponzetto, and Chris Biemann. "TAXI at SemEval-2016 Task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling." In: *Proceedings of SemEval* (2016), pp. 1320–1327.
- [125] zzet Pembeci. "Using Word Embeddings for Ontology Enrichment." In: *International Journal of Intelligent Systems and Applications in Engineering* 4.3 (2016), pp. 49–56.
- [126] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: <http://www.aclweb.org/anthology/D14-1162>.
- [127] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online learning of social representations." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 701–710.
- [128] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." In: *arXiv preprint arXiv:1802.05365* (2018).
- [129] Joel Pocostales. "NUIG-UNLP at SemEval-2016 Task 13: A simple word embedding-based approach for taxonomy extraction." In: *Proceedings of SemEval* (2016), pp. 1298–1302.
- [130] Abinash Pujahari and Vineet Padmanabhan. "An Approach to Content Based Recommender Systems Using Decision List Based Classification with k-DNF Rule Set." In: *Information Technology (ICIT), 2014 International Conference on*. IEEE. 2014, pp. 260–263.
- [131] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
- [132] Jesse Read, Bernhard Pfahringer, and Geoffrey Holmes. "Multi-label classification using ensembles of pruned sets." In: *8th IEEE international conference on data mining*. IEEE. 2008, pp. 995–1000.
- [133] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: an open architecture for collaborative filtering of netnews." In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM. 1994, pp. 175–186.
- [134] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.
- [135] Vala Ali Rohani, Zarinah Mohd Kasirun, and Kuru Ratnavelu. "An Enhanced Content-Based Recommender System for Academic Social Networks." In: *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*. IEEE. 2014, pp. 424–431.

- [136] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. "Automatic keyword extraction from individual documents." In: *Text Mining: Applications and Theory* (2010), pp. 1–20.
- [137] Lukas Schmelzeisen and Steffen Staab. "Learning Taxonomies of Concepts and not Words using Contextualized Word Representations: A Position Paper." English. In: *CoRR abs/1902.02169* (Jan. 2019).
- [138] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks." In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.
- [139] Fabrizio Sebastiani. "Text categorization." In: *Encyclopedia of Database Technologies and Applications*. IGI Global, 2005, pp. 683–687.
- [140] Sagnika Sen, Jie Tao, and Amit V Deokar. "On the role of ontologies in information extraction." In: *Reshaping Society through Analytics, Collaboration, and Decision Support*. Springer, 2015, pp. 115–133.
- [141] Upendra Shardanand and Pattie Maes. "Social information filtering: algorithms for automating "word of mouth"." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co. 1995, pp. 210–217.
- [142] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. "Learning semantic representations using convolutional neural networks for web search." In: *Proceedings of the 23rd International Conference on World Wide Web*. ACM. 2014, pp. 373–374.
- [143] Mohammad S Sorower. "A literature survey on algorithms for multi-label learning." In: *Oregon State University, Corvallis* 18 (2010), pp. 1–25.
- [144] Pascal Soucy and Guy W Mineau. "Beyond TFIDF weighting for text categorization in the vector space model." In: *IJCAI*. Vol. 5. 2005, pp. 1130–1135.
- [145] Robert Speer, Joshua Chin, and Catherine Havasi. "Conceptnet 5.5: An open multilingual graph of general knowledge." In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.
- [146] Robert Speer and Catherine Havasi. "Representing General Relational Knowledge in ConceptNet 5." In: *LREC*. 2012, pp. 3679–3686.
- [147] Newton Spolaôr, Everton Alvares Cherman, Maria Carolina Monard, and Huei Diana Lee. "A comparison of multi-label feature selection methods using the problem transformation approach." In: *Electronic Notes in Theoretical Computer Science* 292 (2013), pp. 135–151.
- [148] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

- [149] Ang Sun, Ralph Grishman, and Satoshi Sekine. "Semi-supervised relation extraction with large-scale word clustering." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 521–529.
- [150] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. "Multi-instance multi-label learning for relation extraction." In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics. 2012, pp. 455–465.
- [151] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.
- [152] Susumu Tamagawa, Shinya Sakurai, Takuya Tejima, Takeshi Morita, Noriaki Izumi, and Takahira Yamaguchi. "Learning a large scale of ontology from Japanese Wikipedia." In: *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 1. IEEE. 2010, pp. 279–286.
- [153] Liling Tan, Francis Bond, and Josef van Genabith. "Usaar at semeval-2016 task 13: Hyponym endocentricity." In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. 2016, pp. 1303–1309.
- [154] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. "Line: Large-scale information network embedding." In: *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee. 2015, pp. 1067–1077.
- [155] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. "Arnet-miner: extraction and mining of academic social networks." In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 990–998.
- [156] Jiliang Tang, Salem Alelyani, and Huan Liu. "Feature selection for classification: A review." In: *Data classification: Algorithms and applications* (2014), p. 37.
- [157] Roberto Torres, Sean M McNee, Mara Abel, Joseph A Konstan, and John Riedl. "Enhancing digital libraries with TechLens+." In: *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2004, pp. 228–236.
- [158] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. "An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition." In: *BMC bioinformatics* 16.1 (2015), p. 138.
- [159] Grigorios Tsoumakas and Ioannis Katakis. "Multi-label classification: An overview." In: *International Journal of Data Warehousing and Mining (IJDWM)* 3.3 (2007), pp. 1–13.

- [160] Grigorios Tsoumakas and Ioannis Vlahavas. "Random k-labelsets: An ensemble method for multilabel classification." In: *European conference on machine learning*. Springer. 2007, pp. 406–417.
- [161] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. "Relational Stacked Denoising Autoencoder for Tag Recommendation." In: *AAAI*. 2015, pp. 3052–3058.
- [162] Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. "Relation classification via multi-level attention cnns." In: (2016).
- [163] Shanchan Wu, Kai Fan, and Qiong Zhang. "Improving Distantly Supervised Relation Extraction with Neural Noise Converter and Conditional Optimal Selector." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 7273–7280.
- [164] Peng Xu and Denilson Barbosa. "Connecting Language and Knowledge with Heterogeneous Representations for Neural Relation Extraction." In: *arXiv preprint arXiv:1903.10126* (2019).
- [165] Zhenghua Xu, Cheng Chen, Thomas Lukasiewicz, Yishu Miao, and Xiangwu Meng. "Tag-aware personalized recommendation using a deep-semantic similarity model with negative sampling." In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM. 2016, pp. 1921–1924.
- [166] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. "Network representation learning with rich text information." In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.
- [167] Chenxing Yang, Baogang Wei, Jiangqin Wu, Yin Zhang, and Liang Zhang. "CARES: a ranking-oriented CADAL recommender system." In: *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2009, pp. 203–212.
- [168] Yiming Yang. "An evaluation of statistical approaches to text categorization." In: *Information retrieval 1.1-2* (1999), pp. 69–90.
- [169] Yiming Yang, Xin Liu, et al. "A re-examination of text categorization methods." In: *Sigir*. Vol. 99. 8. 1999, p. 99.
- [170] Yiming Yang and Jan Pedersen. "A Comparative Study on Feature Selection in Text Categorization." In: Morgan Kaufmann Publishers, 1997, pp. 412–420.
- [171] Yiming Yang and Jan O Pedersen. "A comparative study on feature selection in text categorization." In: *Icml*. Vol. 97. 1997, pp. 412–420.
- [172] Wen-tau Yih, Xiaodong He, and Christopher Meek. "Semantic Parsing for Single-Relation Question Answering." In: *ACL (2)*. Citeseer. 2014, pp. 643–648.
- [173] Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. "Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks." In: *arXiv preprint arXiv:1811.01727* (2018).

- [174] Ronghui You, Zihan Zhang, Suyang Dai, and Shanfeng Zhu. "HAXMLNet: Hierarchical Attention Network for Extreme Multi-Label Text Classification." In: *arXiv preprint arXiv:1904.12578* (2019).
- [175] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. "Recent trends in deep learning based natural language processing." In: *IEEE Computational Intelligence Magazine* 13.3 (2018), pp. 55–75.
- [176] Jianfei Yu, Luis Marujo, Jing Jiang, Pradeep Karuturi, and William Brendel. "Improving multi-label emotion classification via sentiment classification with dual attention transfer network." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 1097–1102.
- [177] Zhiwen Yu, Yuichi Nakamura, Seiie Jang, Shoji Kajita, and Kenji Mase. "Ontology-based semantic recommendation for context-aware e-learning." In: *Ubiquitous Intelligence and Computing* (2007), pp. 898–907.
- [178] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. "Relation Classification via Convolutional Deep Neural Network." In: *COLING*. 2014, pp. 2335–2344.
- [179] Chengxiang Zhai and John Lafferty. "Model-based feedback in the KL-divergence retrieval model." In: *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*. 2001, pp. 403–410.
- [180] Min-Ling Zhang and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning." In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.
- [181] Shuai Zhang, Lina Yao, and Aixin Sun. "Deep learning based recommender system: A survey and new perspectives." In: *arXiv preprint arXiv:1707.07435* (2017).
- [182] Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. "Deep Extreme Multi-label Learning." In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM. 2018, pp. 100–107.
- [183] Xiatian Zhang, Quan Yuan, Shiwan Zhao, Wei Fan, Wentao Zheng, and Zhong Wang. "Multi-label classification without the multi-label cost." In: *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM. 2010, pp. 778–789.
- [184] Ye Zhang, Libin Yang, Xiaoyan Cai, and Hang Dai. "A novel personalized citation recommendation approach based on gan." In: *International Symposium on Methodologies for Intelligent Systems*. Springer. 2018, pp. 268–278.
- [185] Junbo Zhao, Michael Mathieu, and Yann LeCun. "Energy-based generative adversarial network." In: *arXiv preprint arXiv:1609.03126* (2016).
- [186] Peng Zhou and Nora El-Gohary. "Ontology-based multilabel text classification of construction regulatory documents." In: *Journal of Computing in Civil Engineering* 30.4 (2015), p. 04015058.



- [187] Fuzhen Zhuang, Zhiqiang Zhang, Mingda Qian, Chuan Shi, Xing Xie, and Qing He. “Representation learning via Dual-Autoencoder for recommendation.” In: *Neural Networks* 90 (2017), pp. 83–89.

*All web pages cited in this work have been checked in May 2019. However, due to the dynamic nature of the World Wide Web, their long-term availability cannot be guaranteed.*

## APPENDIX

## A.1 PART-OF-SPEECH TAGS

This section gives an overview of the PoS tags used in this dissertation. Table 39 shows the tags that are relevant for English from Penn Treebank Project [0].

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	DT	Determiner
3.	IN	Preposition or subordinating conjunction
4.	JJ	Adjective
5.	JJR	Adjective, comparative
6.	JJS	Adjective, superlative
7.	NN	Noun, singular or mass
8.	NNS	Noun, plural
9.	NNP	Proper noun, singular
10.	NNPS	Proper noun, plural
11.	POS	Possessive ending
12.	PRP	Personal pronoun
13.	PRP\$	Possessive pronoun
14.	RB	Adverb
15.	RBR	Adverb, comparative
16.	RBS	Adverb, superlative
17.	VB	Verb, base form
18.	VBD	Verb, past tense
19.	VBG	Verb, gerund or present participle
20.	VBN	Verb, past participle
21.	VBP	Verb, non-3rd person singular present
22.	VBZ	Verb, 3rd person singular present

Table 39: An overview of the common PoS tags from Penn Treebank Project [0]



## A.2 LIST OF ACRONYMS

ARGA	Adversarial Regularized Graph Autoencoder
BiRNN	Bidirectional RNN
BOW	Bag-of-Words
BR	Binary Relevance
CBF	Content-based Filtering
CBOW	Continuous-Bag-of-Words
CD	Concept Degree
CF	Collaborative Filtering
Ctree	Conditional Inference Tree
DF	Document Frequency
DSSM	Deep Semantic Similarity Model
FDNN	Feed-foward Neural Network
FN	False Negative
FP	False Positve
GAE	Graph Autoencoder
GAN	Generative Adversarial Network
GloVe	Global Vector For Word Representation
GR	Gain Ratio
GRU	Gated Recurrent Unit
HITS	Hyperlink-Induced Topic Search
IG	Information Gain
IR	Information Retrieval
KNN	K-Nearest-Neighbor
LP	Label Powerset
LSTM	Long Short Term Memory
MAP	Mean Average Precision
ML	Machine Learning
ML-DT	Multi-label Decision Tree
ML-KNN	Multi-label K Nearest Neighbors
MRR	Mean Reciprocal Rank
NER	Named Entity Recognition
NLP	Natural Language Processing
NP	Noun Phrase
OER	Open Educational Resources

PoS	Part-of-Speech
RAKE	Rapid Automatic Keyword Extraction
RAKEL	Random K Labelsets
ReLU	Rectified Linear Units
Seq2Seq	Sequence-to-Sequence
SVM	Support Vector Machine
TC	Term Count
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
VGAE	Variational Graph Autoencoder
VRRW	Vertex Reinforced Random Walk

## A.3 SUPERVISED STUDENT THESES

- [1] Yassin Alkhalili. "Using Deep Learning for Categorizing Biomedical Abstract." Master Thesis. TU Darmstadt, 2018.
- [2] Luna Alrawas. "Cross-lingual Multi-label Text Classification." Master Thesis. TU Darmstadt, 2019.
- [3] Eid Araache. "Content-based Recommender System Using Word Embeddings for Recommending Next Learning Step." Master Thesis. TU Darmstadt, 2017.
- [4] Yassine Aziani. "Graph-based Recommender System Using Semantic Link Networks and Word Embeddings." Bachelor Thesis. TU Darmstadt, 2019.
- [5] Anay Deshpande. "Query-based Recommender System Using Deep Learning and Word embeddings." Master Thesis. TU Darmstadt, 2018.
- [6] Letian Feng. "Deep Learning for Recommendation of Resources for the next Learning Step." Master Thesis. TU Darmstadt, 2018.
- [7] Leon Alexander Herrmann. "Identification of Meaningful Syntactic and Semantic Relations between Words in Web Documents based on Word Embeddings." Bachelor Thesis. TU Darmstadt, 2016.
- [8] Noor Isbel. "Collaborating Aesthetic Change and Heterogeneous Information into Recommender Systems Using Deep Learning." Master Thesis. TU Darmstadt, 2019.
- [9] Robin Kahlow. "Deep Learning Techniques for Automatic Extraction and Classification of Semantic Relations from Unstructured Texts." Bachelor Thesis. TU Darmstadt, 2017.
- [10] Hatem Khalloof. "An Extensive Analytical Study of Dataport Electricity Dataset for Accurately Predicting Short- and Long-Term Power Consumption and Generation Trends." Master Thesis. TU Darmstadt, 2016.
- [11] David Langsam. "Personalized Citation Recommendation Using Generative Adversarial Networks." Bachelor Thesis. TU Darmstadt, 2019.
- [12] Junhong Liu. "Efficient Multi-Label Text Classification Using Semantic Word Vectors." Master Thesis. TU Darmstadt, 2017.
- [13] Tobias Michels. "Using Deep Learning for Multi-label Text Classification of Long Text." Master Thesis. TU Darmstadt, 2018.
- [14] Nicolai Minter. "Chatbots as Conversational Recommender Systems in Books Context." Bachelor Thesis. TU Darmstadt, 2019.
- [15] Saba Sabrin. "Ontology-based Multi-label Text Classification." Master Thesis. TU Darmstadt, 2017.
- [16] Mian Muhammad Bilal Shabbir. "Designing a Distributed System for Automatic Ontology Extraction from Unstructured Text." Bachelor Thesis. TU Darmstadt, 2016.

- [17] Feras Tanan. "Ontology-based Training-less Multi-label Text Classification Using Word Embeddings." Master Thesis. TU Darmstadt, 2017.
- [18] Ahmed Trabelsi. "Combining Heterogeneous Information Sources for Product Recommendation Using Generative Adversarial Networks." Master Thesis. TU Darmstadt, 2019.

## AUTHOR'S PUBLICATIONS

## MAIN PUBLICATIONS

- [1] Wael Alkhatib, Alaa Alhamoud, Doreen Böhnstedt, and Ralf Steinmetz. "Hybrid Model for Large Scale Forecasting of Power Consumption." In: *International Work-Conference on Artificial Neural Networks*. Springer. 2017, pp. 661–672.
- [2] Wael Alkhatib, Alaa Alhamoud, Doreen Böhnstedt, and Ralf Steinmetz. "Hybrid Models for Short-Term Load Forecasting Using Clustering and Time Series." In: *International Work-Conference on Artificial Neural Networks*. Springer. 2017, pp. 104–115.
- [3] Wael Alkhatib, Eid Araache, Christoph Rensing, and Steffen Schnitzer. "Ensuring Novelty and Transparency in Learning Resource-Recommendation Based on Deep Learning Techniques." In: *European Conference on Technology Enhanced Learning*. Springer, Cham, 2018, pp. 609–612.
- [4] Wael Alkhatib, Leon Alexander Herrmann, and Christoph Rensing. "Onto.KOM - Towards a Minimally Supervised Ontology Learning System based on Word Embeddings and Convolutional Neural Networks." In: *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. 2017, pp. 17–26.
- [5] Wael Alkhatib and Christoph Rensing. "Towards a Classification of Learning Support Systems at the Digitized Workplace." In: *DeLFI Workshops*. 2016, pp. 188–194.
- [6] Wael Alkhatib and Christoph Rensing. "Personalized Citation Recommendation Using an Ensemble Model of DSSM and Bibliographic Information." In: *(accepted for publication) Advances in Analytics for Learning and Teaching*. Springer. 2019.
- [7] Wael Alkhatib, Christoph Rensing, and Johannes Silberbauer. "Multi-label text classification using semantic features and dimensionality reduction with autoencoders." In: *International Conference on Language, Data and Knowledge*. Springer, Cham. 2017, pp. 380–394.
- [8] Wael Alkhatib, Saba Sabrin, Svenja Neitzel, and Christoph Rensing. "Towards Ontology-Based Training-Less Multi-label Text Classification." In: *International Conference on Applications of Natural Language to Information Systems*. Springer, Cham. 2018, pp. 389–396.



- [9] Wael Alkhatib, Steffen Schnitzer, Wei Ding, Peter Jiang, Yassin Alkhalili, and Christoph Rensing. "Comparison of Feature Selection Techniques for Multi-label Text Classification against a New Semantic-based Method." In: *(Accepted for publication) in the proceeding of the 19th International Conference on Computational Linguistics and Intelligent Text Processing*. 2018.
- [10] Wael Alkhatib, Steffen Schnitzer, and Christoph Rensing. "Training-less Multi-label Text Classification Using Knowledge Bases and Word Embeddings." In: *(accepted for publication) The 12th International Conference on Knowledge Science, Engineering and Management*. 2019.
- [11] Wael Alkhatib, Steffen Schnitzer, Tim Steuer, and Christoph Rensing. "Unsupervised Query-based Document Recommendation Using Deep Learning." In: *In: (Accepted for publication) in the proceeding of the 19th International Conference on Computational Linguistics and Intelligent Text Processing*. Springer, 2019.

## CO-AUTHORED PUBLICATIONS

- [12] Moazzam Butt and Wael Alkhatib. "Robust 2D Face Recognition Under Different Illuminations Using Binarized Partial Face Features: Towards Protecting ID Documents." In: *Computational Forensics*. Springer, 2015, pp. 31–43.
- [13] Naser Damer, Wael Alkhatib, Andreas Braun, and Arjan Kuijper. "Neighbor distance ratios and dynamic weighting in multi-biometric fusion." In: *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, Cham. 2017, pp. 491–500.
- [14] Christoph Rensing, Wael Alkathib, Max Zollenkopf, Olaf Aschmann, et al. "Ein Assistent zur Konzeption von Lernaufträgen für das Lernen am Produktionssarbeitsplatz." In: *Bildungsräume 2017* (2017).
- [15] Steffen Schnitzer, Dominik Reis, Christoph Rensing, Wael Alkhatib, and Ralf Steinmetz. "Preselection of Documents for Personalized Recommendations of Job Postings based on Word Embeddings." In: *(Accepted for Publication) The 34th ACM/SIGAPP Symposium on Applied Computing (SAC 19), April 812, 2019, Limassol, Cyprus*. Limassol, Cyprus: ACM, New York, NY, USA, 2019.

## ACKNOWLEDGEMENTS

---

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr.-Ing. Ralf Steinmetz for giving me the chance to complete my Ph.D. research under his supervision in an exceptionally competitive working environment at KOM. I thank him for the continuous support of my research, and for giving me confidence in my work. My sincere thanks also go to my direct supervisor, PD Dr. Christoph Rensing, for providing guidance and feedback throughout the last 3 years and 8 months of research.

Special thanks go to my parents, my mother Diaa, my father Mohamad, my brothers, and my sisters for supporting me spiritually throughout writing this thesis and my life in general. My father, I'm grateful to you for guiding me to reach new heights and all your words that gave me a perspective which no books can teach me. My mother, there are not enough words to describe how precious and important you are for me. You were always beside me, supporting and caring for me in my life. I am forever grateful.

The last paragraph is dedicated to my wife Nana who has provided me moral and emotional support in my life. Thanks for supporting me whenever I had stress or a problem, and for being patient while I am bringing work home. Every single day you fill happiness into my life. Thank you for coming into my life and make my soul blossom. Thank you, sweetheart.



CURRICULUM VITÆ

---

## PERSONAL

Name	Wael Alkhatib
Date of Birth	October 15, 1988
Place of Birth	Hama, Syria
Nationality	Syrian

## ACADEMIC QUALIFICATION

10/2013-10/2015	Master of Science: Distributed Software Systems, Technical University of Darmstadt Master Thesis: <i>Hybrid Large Scale Power Consumption Prediction</i>
08/2006-11/2011	Eng.Diploma.Information Systems, Higher Institute for Applied Science and Technology, Damas- cus, Syria.

## ACADEMIC EXPERIENCE

Since 04/2016	Researcher funded by BMPF (Federal Ministry of Education and Research) Projekt: KeAP "Kompetenzentwicklung am Arbeitsplatz"
Since 11/2018	Researcher funded by BMPF (Federal Ministry of Education and Research) Projekt: PiobRec "Personal Interest-Oriented Books Recom- mendation"

## WORK EXPERIENCE

Since 04/2016	Technische Universität Darmstadt, Research assistant at the researcher group Knowledge Media at the Multimedia Communications Lab (KOM)
---------------	--

## TEACHING ACTIVITY

2016-2019	Lecture "Communication Networks 1", Teaching Assistant.
-----------	---

- 2016-2019 Lab “Current Topics in Web Applications, Information Management, and Semantics”, Supervisor.
- 2016-2019 Lab “Multimedia Communications Lab/Project”, Supervisor.

## HONORS

- 04/2017 Best Paper Award for “Onto.KOM - Towards a Minimally Supervised Ontology Learning System based on Word Embeddings and Convolutional Neural Networks” at KEOD 2017.
- 07/2017 Best Master Thesis Energy Award for “Hybrid Large Scale Power Consumption Prediction”, Energy Center, TU Darmstadt 2017.
- 10/2018 Software Campus Grant for “Project PIOBRec: Personal Interest-Oriented Books Recommendation”, Software Campus 2018 program.

*Darmstadt, January 28th, 2020*

---

Wael Alkhatib

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

---

ICH versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

*Darmstadt, 28. January 2020*

---

Wael Alkhatib



## COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both  $\text{\LaTeX}$  and  $\text{\LyX}$ :

<https://bitbucket.org/amiede/classicthesis/>

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>

*Final Version* as of June 20, 2020 (`classicthesis`).